System Identification and its Applications, with Emphasis on Direction-dependent Processes

Ai Hui Tan

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

University of Warwick
School of Engineering

March 2002

To my parents and sister

Table of Contents

Acknowledgements

I would like to thank my academic supervisor Professor K. R. Godfrey for his invaluable support and guidance during my Ph.D. studies.

I would also like to thank Professor H. A. Barker of the School of Electrical and Electronic Engineering, University of Wales Swansea and Professors J. Schoukens and R. Pintelon of the Department of Electrical Engineering, Vrije Universiteit Brussel for their stimulating ideas and helpful advice.

Financial support in the forms of an Overseas Research Student Award, from the Committee of Vice-Chancellors and Principals of the Universities of the United Kingdom, and a University Graduate Award (Special Research Studentship) from the University of Warwick is gratefully acknowledged.

Finally, I wish to express my sincere gratitude to my family, friends and colleagues for all their support and encouragement.

Declaration

Material from the following papers has been included in the thesis.

Journal Papers

1. TAN, A. H. and GODFREY, K. R. : 'Identification of processes with direction-dependent dynamics', *IEE Proc. - Control Theory Appl.*, 2001, **148** (5), pp. 362 - 369.

2. TAN, A. H. and GODFREY, K. R. : 'The generation of binary and near-binary pseudo-random signals: an overview', accepted for *IEEE Transactions - Instrum. Meas..*

3. TAN, A. H. and GODFREY, K. R. : 'Identification of Wiener-Hammerstein models using linear interpolation in the frequency domain (LIFRED)', accepted for *IEEE Transactions - Instrum. Meas..*

Conference Papers

1. TAN, A. H. and GODFREY, K. R. : 'Design and application of a new MATLAB routine to generate perturbation signals', *UKACC International Conference 'Control 2000'*, Cambridge, UK, 4 - 7 September 2000, Session 2A, Paper 2.

2. BARKER, H. A., GODFREY, K. R. and TAN, A. H. : 'Identification of systems with direction-dependent dynamics', *Proc. 39th IEEE Conference on Decision and Control (CDC 2000)*, Sydney, Australia, 12 - 15 December 2000, pp. 2843 - 2848.

3. TAN, A. H. and GODFREY, K. R. : 'The generation of binary and near-binary pseudo-random signals: an overview', *Proc. 18th IEEE Instrumentation and Measurement Technology Conference (IMTC)*, Budapest, Hungary, 21 - 23 May 2001, pp. 766 - 771.

4. BARKER, H. A., GODFREY, K. R. and TAN, A. H. : 'Optimised Wiener models for direction-dependent dynamic systems' *Proc. American Control Conference (ACC 2001)*, Arlington, USA, 25 - 27 June 2001, pp. 4880 - 4881.

5. TAN, A. H. and GODFREY, K. R. : 'Modelling of direction-dependent dynamic processes: a comparison of Wiener models and neural networks', accepted for *IEEE Instrumentation and Measurement Technology Conference (IMTC) 2002.*

6. BARKER, H. A., TAN, A. H. and GODFREY, K. R. : 'Wiener models of direction-dependent dynamic systems', accepted for *International Federation of Automatic Control (IFAC) Congress 2002.*

Summary of the Thesis

In the first sub-section of the thesis, signal design for both linear and nonlinear system identification is considered. To identify a linear system using a perturbation test, a binary signal is sufficient and has the advantage of maximising the power available within a specified peak-to-peak amplitude. For this purpose, a program was written to generate five classes of binary and near-binary signal. However, to identify a nonlinear system with a Hammerstein structure, a multi-level signal is required, and methods to optimise such a signal are proposed.

In the second sub-section, the detection of the departure from linearity for direction-dependent processes is considered. It was found that only signals based on maximum length sequences allow the detection of such characteristics due to the coherent patterns formed in the crosscorrelation function. The 'combined' linear dynamics of the system are identified. The modelling of such processes using Wiener and neural network models is investigated. Practical results from an electronic nose are presented. The control of direction-dependent processes using the PID controller is then examined, with the design rules set according to the identified 'combined' dynamics.

The thesis then moves on to the topic of autotuning. The autotuning of Smith predictors for processes with significant dead time is considered. The frequency response of the process is identified in closed-loop using a multisine signal. Tuning rules for robust control are suggested which relate the controller parameters to the process parameters. A real application using a hot-air flow device is illustrated.

The final sub-section of the thesis looks at the identification of Wiener-Hammerstein models. A new technique using linear interpolation is proposed which is based on the symmetry properties of the Volterra kernel. This method has the advantages that only a single experiment is needed, and it is simple to use since no optimisation or recursive computations are required. Simulation examples are provided to illustrate the effectiveness of the technique, and its robustness in the presence of noise and input signal distortion.

Glossary of Abbreviations Used

AIC	Akaike's information criterion
ARX	autoregressive model with exogenous input
ARMAX	autoregressive moving average model with exogenous input
ELiS	Estimator for Linear Systems
DFT	discrete Fourier transform
EMINE	effective minimum ratio of actual to specified harmonic amplitude
GF	Galois field
HAB	Hall binary
LIFRED	linear interpolation in the frequency domain
MLB	maximum length binary
MLML	multi-level maximum length
MOS	metal oxide semiconductor
MSE	mean squared error
NID	no interharmonic distortion
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PIPS	Performance Index for Perturbation Signals
PIPSE	Effective Performance Index for Perturbation Signals
PRB	pseudo-random binary
QRB	quadratic residue binary
QRT	quadratic residue ternary
SNR	signal-to-noise ratio
TPB	Twin Prime binary

Chapter 1

Introduction

Many processes in the real world are complex in structure, with their static and dynamic characteristics being influenced by several physical quantities which may or may not be easily measured or controlled. In such situations, a simplified model of the process is necessary in order that the process can be simulated and studied. Identification is a powerful tool to achieve this end and indeed, this has been the topic of several decades of research, partly because it is a very broad subject, which covers areas such as experimental design, model construction and parameter estimation.

The importance of experimental design cannot be overstated. It provides the basis for the latter stages in the identification process. A poorly designed experiment more often than not leads to poor results, and has to be redesigned and repeated. To avoid this, several aspects of the experiment must be carefully considered. The main ones include the type of perturbation signal used and the frequencies of interest, with the latter closely linked to the system bandwidth. The 'best' type of perturbation signal to apply depends on the particular application and the information which is to be extracted from the experiment. However, in most circumstances, periodic signals should be used because of their many advantages. These include the possibility of averaging several periods of measurements to improve the signal-to-noise ratio, the characterisation of the noise and nonlinearities present through a careful selection of the input power spectrum, the elimination of leakage errors in the discrete Fourier transform when an integer number of periods is measured, the elimination of non-excited frequencies resulting in data reduction, and the detection of trends (such as drift) in the process. Hence, in Chapter 2, the design of periodic pseudo-random signals is considered, both for the identification of linear and nonlinear systems. In fact, throughout the thesis, only periodic signals are used, due to the reasons given above.

Chapters 3 to 7 concern a class of processes known as direction-dependent processes. These processes have unsymmetrical properties depending on the direction of the output (whether the output is increasing or decreasing). Such characteristics are detected through the use of maximum length binary (MLB) signals, and the dynamics are identified both in the time and frequency domains. The processes are then modelled

using two different models - the first being a block-oriented structure known as the Wiener model, and the second being a neural network. As an application, these processes are controlled based on their 'combined' linear dynamics using a standard Proportional-Integral-Derivative (PID) controller. The interaction between the aspects of experimental design, model construction and parameter estimation can be clearly seen in this sub-section of the thesis.

In Chapter 8, the control of processes with significant dead time is investigated. These processes are known to be very difficult to control as the dead time could cause problems with stability. They are quite common in industry, mainly due to transport lag. Examples are distillation processes where the liquid being distilled has to travel the length of the distillation column, and coupled-tank systems where the liquid has to travel through a distance of the connecting pipe. The manufacturing of metals, plastics and rubber in sheet form using rolling processes is another well known example. In this case, the controller will change the sheet thickness by adjusting the separation between the rolls. However, a time delay is often introduced by the thickness sensor being located some distance 'downstream' of the process. In Chapter 8, a Smith predictor is used to compensate for the dead time, and the autotuning of the PI controller (with a Smith predictor added) is investigated. To do this, such processes are first identified in closed-loop using a multisine signal which has the advantage of being able to completely meet the frequency domain specification.

The multisine signal is also made use of in Chapters 9 and 10 where a novel approach to identify a Wiener-Hammerstein model is proposed. This is a block-oriented structure with a nonlinear block sandwiched between two blocks of linear subsystems. Such a structure is very popular in practice due to its simplicity compared to the direct use of Volterra series expansions and the fact that it can model many nonlinear processes reasonably well, with the exceptions of those with hysteresis, dead-zone and backlash. The linear subsystems are estimated from the information of the Volterra kernels. This is achieved through the extraction of the symmetry properties of the kernel and the estimation using linear interpolation in the frequency domain. The method is proposed

for a Wiener-Hammerstein model with a quadratic nonlinearity (in Chapter 9) and then extended to that with a cubic nonlinearity (in Chapter 10).

Chapters 2 and 8 of the thesis can be read independently of any other material. However, Chapters 3 to 7 should be considered together as all these involve processes with direction-dependent characteristics. Despite the fact that these chapters appear to consider very different aspects of such processes, similar concepts are used throughout. Also, the applicability of some of the techniques suggested will only become clear when these chapters are read together. Similarly, Chapters 9 and 10 should be considered as a coherent sub-section of the thesis since Chapter 10 is a direct extension of the technique proposed in Chapter 9.

Even though the thesis considers the identification of several different processes, using several different techniques, the methods proposed can all be used in a coherent manner, which is what binds the chapters in the thesis together. In fact, the different choices of perturbation signals (for example, binary or multi-level), frequency domain specifications (for example, all harmonics present or only odd harmonics present) and estimation techniques (for example, time domain estimation or frequency domain estimation) clearly bring out the important aspects of identification, and emphasise those that have a great impact on the success or failure of the identification process as a whole. A summary of the contents of each chapter now follows.

Chapter 2 considers an important aspect of experimental design, which is the design of perturbation signals. This is of paramount importance in the field of system identification as a poorly designed signal will lead to poor measurements and hence result in inaccurate and insufficient knowledge of the system. To identify a linear system using a perturbation test, a binary signal is sufficient and has the advantage of maximising the power available within a specified peak-to-peak amplitude. A program to design binary and near-binary pseudo-random signals was written to generate five classes of signal which are the maximum length binary (MLB) signal, quadratic residue binary (QRB) signal, Hall binary (HAB) signal, Twin Prime binary (TPB) signal and

quadratic residue ternary (QRT) signal. The QRT signal is very close to binary (provided the signal is reasonably long) since only one element in a period is at signal level zero (with the other two levels at +V and -V).

Three measures of signal quality are discussed. The first two of these, the Performance Index for Perturbation Signals (PIPS) and the Effective Performance Index for Perturbation Signals (PIPSE) measure the performance of a signal in maximising the power in the specified harmonics within amplitude constraints. This is important since the signal should be small enough to negate the effects of nonlinearities but large enough to negate the effects of noise. The third measure is the effective minimum ratio between the actual harmonic amplitude and the specified harmonic amplitude at any of the specified harmonics, EMINE. A good signal design is a compromise between maximising PIPS (or PIPSE) and EMINE as these have conflicting requirements.

While a binary signal is the best signal to use in identifying a linear system and a nonlinear model with a Wiener structure (since the linear block precedes the nonlinear block), this is not the case for a Hammerstein model (where the nonlinear block precedes the linear block). A multi-level signal must be used and the number of signal levels must be larger than the order of the nonlinearity. Some properties of such a pseudo-random signal generated from the Galois field are discussed. The optimal signal levels and the number of occurrences of each level are investigated through describing the system using a Vandermonde matrix. Four measures are used to evaluate the performance of the matrix and these involve the sum of all sensitivities in solving for the coefficients of the nonlinearity in the presence of noise, its greatest sensitivity, the norm of the inverse Vandermonde matrix, and the 2-norm condition number of the Vandermonde matrix.

In Chapters 3 to 7, processes with direction-dependent dynamics are considered. This means that the dynamics are different depending on whether the output is increasing or decreasing. Examples of such processes in the industry include steam-raising plants, gas turbines, chemical processes, nuclear reactors, distillation columns, polymerisation

processes, automotive suspensions and tyres. In Chapter 3, the detection of the departure from linearity is considered. It is found that only signals based on maximum length sequences allow the detection of such characteristics due to the coherent patterns formed in the crosscorrelation function. These patterns are caused by the shift-and-add and shift-and-subtract properties of these signals. After detecting the departure from linearity, the 'combined' linear dynamics of the system are identified in Chapter 4 using correlation analysis, and various models and estimators (such as the autoregressive (ARX) and the autoregressive moving average (ARMAX) models, and the Estimator for Linear Systems (ELiS)) in the System Identification Toolbox and the Frequency Domain System Identification Toolbox in MATLAB. It is concluded that while signals based on maximum length sequences should be used for the detection of the departure from linearity, their corresponding inverse-repeat signals should be used for estimating the linear dynamics. This is because the use of inverse-repeat signals eliminates the effects of even order nonlinearities, hence resulting in higher accuracy in the estimates obtained.

The direction-dependent behaviour is inherently very difficult to model due to the nonlinearity being non-static. In Chapter 5, the Wiener model, in which the linear subsystem precedes the nonlinearity, is found to be a good model for such a process. The crosscorrelation function is derived in detail for a first order direction-dependent process and its corresponding Wiener model when the input perturbation is an MLB signal or an inverse-repeat MLB signal. The optimum parameters of the Wiener model are obtained using the Optimization Toolbox in MATLAB. Unfortunately, the theory cannot be extended to second order processes, and instead, simulation results are presented. It is found that the match between the direction-dependent and Wiener processes is not as good as in the first order case, except when the direction-dependent process has a large damping factor and fast dynamics and therefore behaves like a first order process. In such cases, a Wiener model with first order dynamics can be used instead.

In the thesis, the modelling of such processes using a neural network is also considered, and this is the main topic of Chapter 6. The network has an architecture which is based on, and modified from, the fully recurrent network. Such an architecture is referred to as 'semirecurrent' since only past values of the predictions of the network are fed back to the input layer. The performances of the two types of models (the Wiener model and the neural network model) are compared for different input signals and process dynamics. An experiment is conducted on a real process, namely an electronic nose. The input to the process is the chemical odour (acetone in this case) and this is controlled by a set of valves which determine if the odour could reach the metal oxide semiconductor (MOS) sensor. The output of the process is the voltage across the sensor. The direction-dependent characteristics of the electronic nose is caused by the different adsorption and desorption rates of the sensor. It is found that the neural network model works better in this case due to the process being predominantly first order and the input signal being binary. This is consistent with simulated results. However, the Wiener model provides a better match for an actual process when a multisine input is used and the process is first order. Also, when the process is second order, it is found that only the Wiener model is capable of modelling the process.

The control of direction-dependent processes using a PID controller is considered in Chapter 7. While such processes are normally controlled using two different sets of controller parameters depending on the direction of the output, design rules based on the identified 'combined' linear dynamics are studied in this chapter. The aim is to investigate the controller performance when a single set of controller parameters is used. When the process gains in the upward and downward directions are different, limit cycles may occur. (However, for most processes, the gain is the same in the two directions and the difference is only in the time constants.) Controller performance is improved by including a moving averager between the controller and the process. This smoothes the control signal and the process output, thus avoiding the occurrence of limit cycles.

The thesis then moves on to the topic of autotuning in Chapter 8. The autotuning of Smith predictors for processes with significant dead time is considered. The frequency response of the process is identified in closed-loop using a multisine signal. Next, a parametric model is obtained using least squares. Tuning rules for robust control are suggested which relate the controller parameters to the process parameters. This reduces the problem of autotuning to that of a relatively simple task of process identification. Reduced order modelling for high order processes and for those with non-minimum phase is also studied. These processes can be modelled using a lower order model with dead time. An experiment was carried out using a hot-air flow device which proves that the performance of the Smith predictor with PI control is better than that either using the PI control alone, or using the Dahlin controller. This is indeed a very interesting result considering that the ratio of the dead time to the time constants of the process is in fact less than unity.

The final part of the thesis looks at the identification of Wiener-Hammerstein models. In Chapter 9, a new technique using linear interpolation in the frequency domain (shortened as LIFRED) is proposed for the identification of such a process with quadratic nonlinearity. The idea is based on the extraction of the symmetry properties of the Volterra kernel. This method, which makes use of multisine signals with no interharmonic distortion, has the advantages that only a single experiment is needed, and it is simple to use since no optimisation or recursive computations are required. Simulation examples are provided to illustrate the effectiveness of the technique, and its robustness in the presence of noise and input signal distortion. It is shown that despite the use of linear interpolation, the method works well even if the frequency response gain and phase curves could not be approximated by a straight line.

In Chapter 10, the method is extended to a Wiener-Hammerstein process with cubic nonlinearity. It is found that in this case, input signal distortion can cause major problems in the measurement of the Volterra kernel. This is due to the large number of possible combinations of the input harmonics which contribute to a particular harmonic in the output. It is also shown that the estimation lines in the output which are at the

same frequencies as the input harmonics can be used in the estimation of the gain of the first linear subsystem (the one that precedes the nonlinearity). These are normally discarded in most estimation techniques due to the contributions of several different combination of the input frequencies falling on each of these estimation lines, and hence cannot be separated. However, these lines have a larger magnitude (due to the contribution of the linear term) and it is therefore advantageous to make full use of them in the estimation process, particularly since they are always present and could not be eliminated by signal design.

Chapter 11 summarises the main conclusions and highlights the main contributions of the thesis. Suggestions for further research are proposed. It is hoped that the ideas put forward and the many examples illustrated will give the reader a broader perspective of several important aspects of system identification.

Chapter 2

Design of Pseudo-random Perturbation Signals for System Identification

2.1 Abstract

Pseudo-random signals have been widely used for the purpose of system identification. To identify a linear system or a nonlinear system with a Wiener structure, a binary signal is preferred as the signal power is maximised within constraints of the signal amplitude. Among the classes of pseudo-random binary signals, those based on maximum length sequences are the best known because of their ease of generation using feedback shift registers. However, it is less well known that there are several other classes of binary and near-binary signal with identical, or nearly identical, autocorrelation properties. An overview of these classes of signal is given and the design of a new MATLAB routine incorporating all these classes of signal (including the maximum length binary signal) is described. Three measures of signal quality are also reviewed. This chapter then goes on to consider the design of multi-level signals for the identification of nonlinear systems with the Hammerstein structure. Four measures related to the Vandermonde matrix of the system are used to evaluate the optimal signal levels and the number of occurrences of each of these levels.

2.2 Introduction

Periodic signals [1, 2] have been widely used in the field of system identification. These signals can be split into two main categories. The first is computer-optimised signals. Examples are multisine (sum of harmonics) signals [3] and discrete interval binary signals [4] which can be generated using the MATLAB Frequency Domain System Identification Toolbox [5], and multi-level multi-harmonic signals [6].

The second category of periodic signals is pseudo-random signals, which have fixed power spectra (for a given signal length). The best known class is maximum length binary (MLB) [7] signals. In addition, maximum length multi-level signals can be generated using the package GALOIS [8].

Appropriately chosen pseudo-random signals provide highly acceptable alternatives to multisine signals in applications requiring uniform power in the frequency spectrum [4]. This is the most common requirement in practice. In many cases, the input transducer of a system can only cope with a small number of discrete amplitudes, resulting in a problem with the use of multisines, which effectively have an infinite number of levels. An additional advantage of binary signals is that, within constraints of the signal amplitude, the power injected into the process is maximised (or nearly maximised in the case of near-binary signals) which means that the effects of noise can be minimised. This makes binary signals the preferred choice in the identification of linear systems. The only disadvantage of pseudo-random signals compared with multisines is that non-uniform specifications in the frequency spectrum cannot be exactly met as their frequency spectrum is fixed.

MLB signals have periods $N = 2^n - 1$ (where n is an integer > 1) [9], that is $N = 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047$, etc. and can be easily generated in hardware using shift registers. The shape of the autocorrelation function approximates that of an impulse. For an MLB signal with levels ±1, the on-peak value of the autocorrelation is +1 and the off-peak value is $-1/N$. However, as N increases, there are large gaps between successive periods [4]. This may pose a problem if an inflexible specification is given since the limited number of possible periods may result in the use of a signal with an excessively large value of N, so that the specified harmonics contain too little of the total signal power [4]. Fortunately, these gaps can be filled in by other classes of binary and near-binary signal such as quadratic residue binary (QRB), Hall binary (HAB) and Twin Prime binary (TPB) signals, which have identical autocorrelation properties to MLB signals, and with quadratic residue ternary (QRT) signals, which have almost identical autocorrelation functions [10]. It is therefore useful to design a new MATLAB routine to generate all these classes of signals, and this will be discussed in Section 2.3.

Some measures of signal quality for linear system identification will also be considered. These measures have several advantages compared with some measures used in the past such as the Peak Factor, Crest Factor and Form Factor. The ideas used for the identification of linear systems can be extended to the identification of nonlinear

systems, such as the Wiener and Hammerstein structures. These structures have been widely investigated in the literature, for example in [11 - 16]. The use of pseudo-random binary signals for the identification of nonlinear systems is advocated in [17], and this is indeed true for nonlinear systems with the Wiener structure since the signal is a direct input to the linear subsystem [18]. However, binary signals do not sufficiently excite Hammerstein systems and this is shown in [19, 20] using quadratic Volterra filters. The design of multi-level signals for the optimum identification of nonlinear systems with the Hammerstein structure is considered in Section 2.4. Four measures are used to evaluate, in terms of obtaining the maximum accuracy in solving for the coefficients of the nonlinearity in the presence of noise, the quality of the Vandermonde matrix describing the system. It will be shown that the optimal signal in the above context is contradictory to the requirement of maximising the signal power within a specified signal amplitude.

2.3 Design of a New MATLAB Routine

2.3.1 Generation of Maximum Length Binary Signals and the Design of Related Functions

2.3.1.1 Generation of Maximum Length Binary Signals

MLB signal exists for lengths $N = 2^n - 1$, where n is an integer > 1. They are generated in hardware using shift registers consisting of n stages. The feedback to the first stage is the modulo-2 sum of the logic value of the last stage and one or more of the other stages. However, not all feedback connections result in an MLB signal. The characteristic equation in the delays D in the shift register must be irreducible and primitive [9]. A polynomial of degree n is irreducible if it is not divisible by any polynomial of degree less than n but greater than zero [9, 10]. This polynomial is primitive if and only if it does not divide $x^r \oplus_q 1$ for any $r < q^n - 1$, where q is the Galois field (which is two in this case) [9, 10]. (The symbol $\oplus_q$ represents modulo-q addition and will be used throughout the thesis.) Tables of all irreducible polynomials

(modulo-2) of degree 16 or less are given in [21] and the polynomials which are primitive are clearly indicated. Similar tables for degree 19 or less are given in [22]. An example of a primitive polynomial (modulo-2) for each n which satisfies $2 \leq n \leq 100$, and for $n = 107$ and 127 is listed in [23].

If the polynomial $c_n x^n \oplus_2 c_{n-1} x^{n-1} \oplus_2 \ldots \oplus_2 c_1 x \oplus_2 c_0 = 0$ is primitive, the characteristic equation is in the delays introduced by the shift register [24] :-

$$c_n D^n \oplus_2 c_{n-1} D^{n-1} \oplus_2 \ldots \oplus_2 c_1 D \oplus_2 c_0 = 0 \tag{2.1}$$

Using the fact that modulo-2 addition is equivalent to modulo-2 subtraction, the feedback configuration [9] is given by

$$c_0 X = c_1 DX \oplus_2 \ldots \oplus_2 c_{n-1} D^{n-1} X \oplus_2 c_n D^n X \tag{2.2}$$

where X is the input signal to the shift register, DX is the sequence at the output of the first stage of the register and so on, so that $D^n X$ is the sequence at the output of the last stage of the n-stage register.

The MATLAB routine allows the user to specify a characteristic equation. If this is not irreducible and primitive, an error message is given. If no characteristic equation is specified, the default MLB signal is generated which makes use of the function *mlbs* [5] in the MATLAB Frequency Domain System Identification Toolbox.

2.3.1.2 Calculation of Resulting Shift due to Shift-and-add Property

The shift-and-add property [25, 26] states that when an MLB signal which is delayed by α is added to the same signal delayed by β, the resulting signal is the same signal delayed by γ. This property is unique to maximum length signals. For sequence levels 0 and 1 :-

$$s_{i-\alpha} \oplus_2 s_{i-\beta} = s_{i-\gamma} \tag{2.3A}$$

where s_i denotes the MLB sequence and i, α, β and γ are integers. When the elements in the sequence are mapped into real numbers ± 1, this becomes the shift-and-multiply property

$$u_{t-\alpha}u_{t-\beta} = -u_{t-\gamma} \tag{2.3B}$$

Thus terms with multiple products of inputs u can be replaced by terms with single lags in u [27]. These lags are dependent on the particular MLB signal used and are different for different MLB signals of the same period.

The MATLAB routine offers functions to calculate positions of the resulting lags for different shifts of the original signal, when two or three (shifted) signals are added together. The first signal is used as a reference and is not shifted. In the case of two signals added together, the second signal is delayed with respect to the first signal. If three signals are added together, the second and third signals are delayed with respect to the first signal. If an inverse-repeat signal is used, the routine allows only three signals to be added as the addition of two signals will not result in a shifted version of the original signal.

2.3.1.3 Measures of Rejection of Unsystematic Errors

When an MLB signal is used to estimate a system weighting function obtained by input-output crosscorrelation, there are errors introduced when nonlinearities that may be described by a Volterra functional series [28] are present [29]. Examples in the literature where these effects are investigated are given in [25, 30]. These errors can be classified into systematic and unsystematic errors. Unsystematic errors are dependent on the particular MLB signal used.

Unsystematic errors due to second order nonlinearities occur when

$$s_{i-J} \oplus_2 s_{i-K} \oplus_2 s_{i-I} = 0 \tag{2.4}$$

where i, J, K and I are integers. Similarly, those due to third order nonlinearities occur when

$$s_{i-J} \oplus_2 s_{i-K} \oplus_2 s_{i-L} \oplus_2 s_{i-I} = 0 \tag{2.5}$$

where L is also an integer.

If $0 \leq J < K < I$ in equation (2.4) and $0 \leq J < K < L < I$ in equation (2.5), two quantities R_0 and R_1 can be defined as the upper bounds of I below which equations (2.4) and (2.5) respectively are not satisfied [29]. These measures can be used as a guide to the position of the nearest significant unsystematic error from the zero-lag position in the crosscorrelation function. Hence, for a particular signal period N, the signal with feedback connections resulting in the largest possible values of R_0 and R_1 is preferred.

MLB signals with the greatest rejection of unsystematic errors and their corresponding values of R_0 and R_1 are tabulated in [29] for $2 \leq n \leq 11$. The values of R_0 and R_1 are calculated and automatically displayed when the MATLAB routine is used to generate an MLB signal. If an inverse-repeat signal is generated, only the value of R_1 will be displayed since the effects of second order nonlinearities are eliminated.

2.3.2 Other Classes of Signal Available

The main objective of the new MATLAB routine is to generate the QRB, QRT, HAB and TPB signals, and their corresponding inverse-repeat signals.

QRB signals exist for $N = 4k - 1$, where k is an integer and N is prime [9, 10], that is $N =$ 3, 7, 11, 19, 23, 31, 43, 47, 59, 67, 71, 79, etc. The sequence $\{x_r\}$, $r = 1, 2, \ldots, N$ is formed from the rule

$$x_r = +1 \text{ if } r \text{ is a square, modulo-}N$$
$$x_r = -1 \text{ otherwise}$$
$$x_N = +1 \text{ or } -1 \qquad (2.6)$$

QRT signals exist for $N = 4k \pm 1$, where k is an integer and N is prime, that is $N =$ 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, etc. Note that this is the most common class of pseudo-random signals in terms of periods available. These are generated using the same formula as for QRB signals except that x_N is set to 0, resulting in a ternary signal with $(N-1)/2$ elements +1, $(N-1)/2$ elements -1 and 1 element 0. The autocorrelation function of a QRT signal is nearly identical to that of an MLB signal and for a QRT signal with

signal levels +1, 0 and -1, the on-peak value of the autocorrelation is $(N\text{-}1)/N$ and the off-peak value is $-1/N$.

HAB signals exist for periods $N = 4k^2 + 27$, where k is an integer and N is prime [10], that is $N = 31, 43, 127, 223, 283, 811, 1051, 1471, 1627$, etc. A primitive root u of N is first chosen. The sequence is formed from the rule that

$$x_r = +1 \text{ if } r \equiv u^t, \text{ modulo-}N, \text{ where } t \equiv 0, 1 \text{ or } 3 \text{ (modulo-6)}$$
$$x_r = -1 \text{ otherwise} \qquad (2.7)$$

A list of least primitive roots up to 4000 is given in [31]. For primes larger than 4000, a separate routine was written to find the least primitive roots and the results were incorporated into the MATLAB routine.

TPB signals exist for $N = k\,(k+2)$, where k and $k+2$ are both primes [10], that is $N = 15, 35, 143, 323, 899, 1763, 3599, 5183$, etc. First QRB sequences are generated for both lengths k and $k+2$; these sequences are denoted by $\{a_r\}$ and $\{b_r\}$ respectively. The TPB sequence $\{x_r\}$ is then defined by

$$x_r = a_r b_r \text{ for } r \neq 0, \text{ modulo-}k \text{ or modulo-}(k+2)$$
$$x_r = +1 \text{ if } r = 0 \text{ modulo-}(k+2)$$
$$x_r = -1 \text{ if } r = 0 \text{ modulo-}k, \text{ but } r \neq 0 \text{ modulo-}(k+2) \qquad (2.8)$$

An alternative method to generate a TPB sequence is to make use of a common primitive root u of k and $(k+2)$ [10]. A difference set is formed, modulo-$k(k+2)$, from

$$1, u, u^2, \ldots, u^{(p^2-3)/2}; \quad 0, (k+2), 2(k+2), \ldots, (p-1)(k+2)$$

A TPB sequence is then derived from the difference set.

The first method is used in the MATLAB routine as it allows the shared use of the function to generate QRB signals.

The MATLAB routine can generate signals up to a length of 50000. This length is limited by the speed in generating TPB signals and also by the accuracy in the calculation of remainders required for the generation of HAB signals.

Two harmonic specifications are available. All the signals above have all harmonics present. Signals with only odd harmonics present can be generated by inverting every other element of the sequence on which the signal is based, so producing an inverse-repeat signal of period $2N$. These signals have the property that the effects of odd and even nonlinearities can be separated at the system output. Additionally, the effects of even order nonlinearities are eliminated in the input-output crosscorrelation function.

2.3.3 Other Functions Available

The program allows the user to plot graphs of the signal, its discrete Fourier transform, power spectrum and autocorrelation function. Examples are given in Figures 2.1 to 2.4 for a TPB signal of length 35. From Figure 2.2, it can be seen that the signal contains all harmonics, and the spectrum is flat except at dc. In Figure 2.3, the power spectrum for three periods is plotted and this has a $\left(\frac{\sin x}{x}\right)^2$ envelope. The on-peak value of the autocorrelation is +1 (at zero delay) and the off-peak value is -1/35. Examples are given in Figures 2.5 to 2.8 for the corresponding inverse-repeat signal. From Figure 2.6, the signal contains only odd harmonics, and the spectrum is flat except at harmonic 35, which is the Nyquist frequency. The autocorrelation function has a value of +1 at zero delay, -1 at a delay of 35 clock-pulse intervals, +1/35 at odd numbers of delay/clock-pulse interval and -1/35 at even numbers of delay/clock-pulse interval.

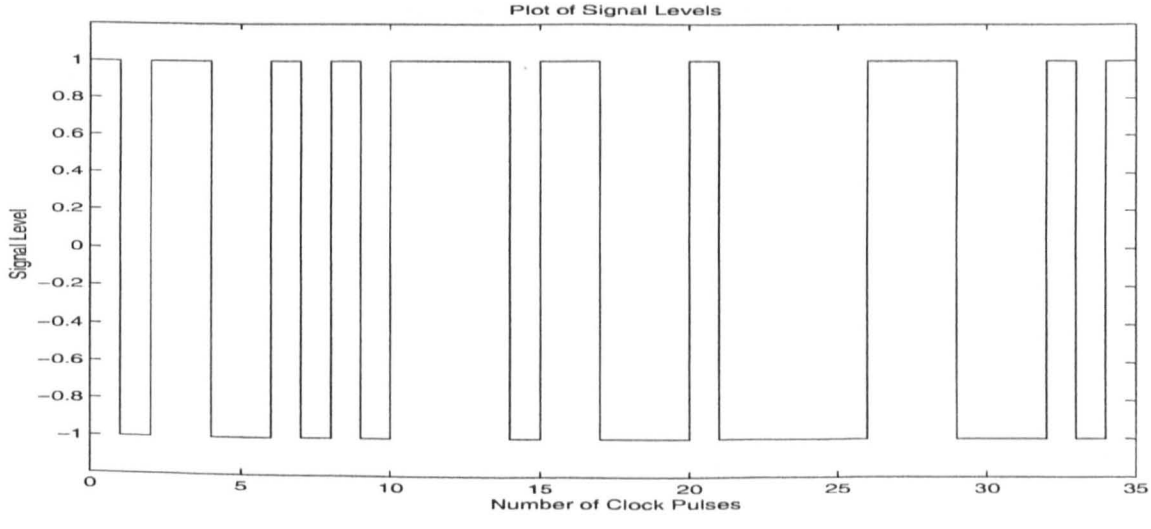

Figure 2.1. TPB signal of length 35.

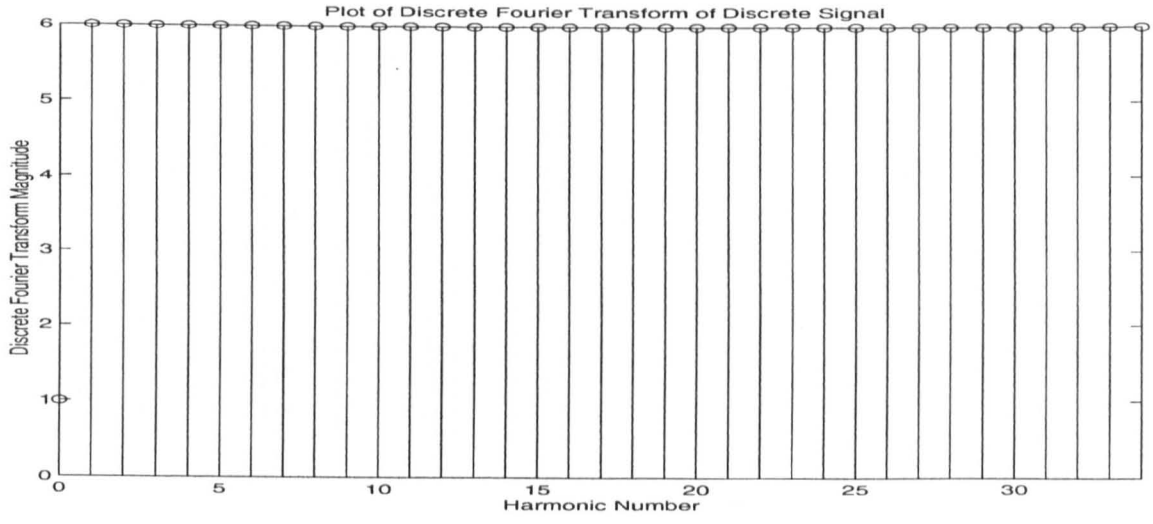

Figure 2.2. Discrete Fourier transform of a TPB signal of length 35.

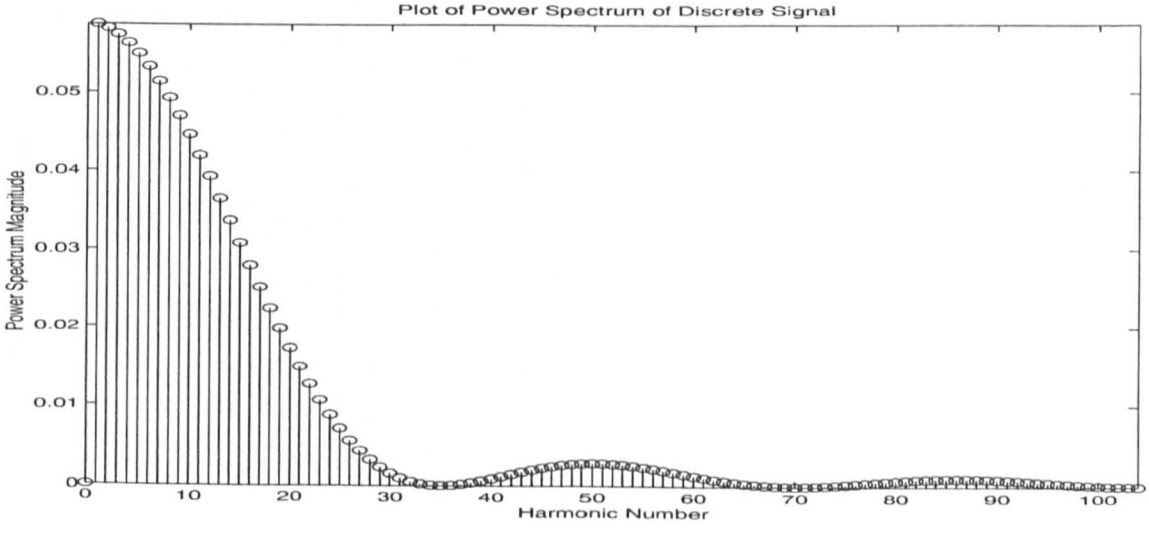

Figure 2.3. Power spectrum of a TPB signal of length 35.

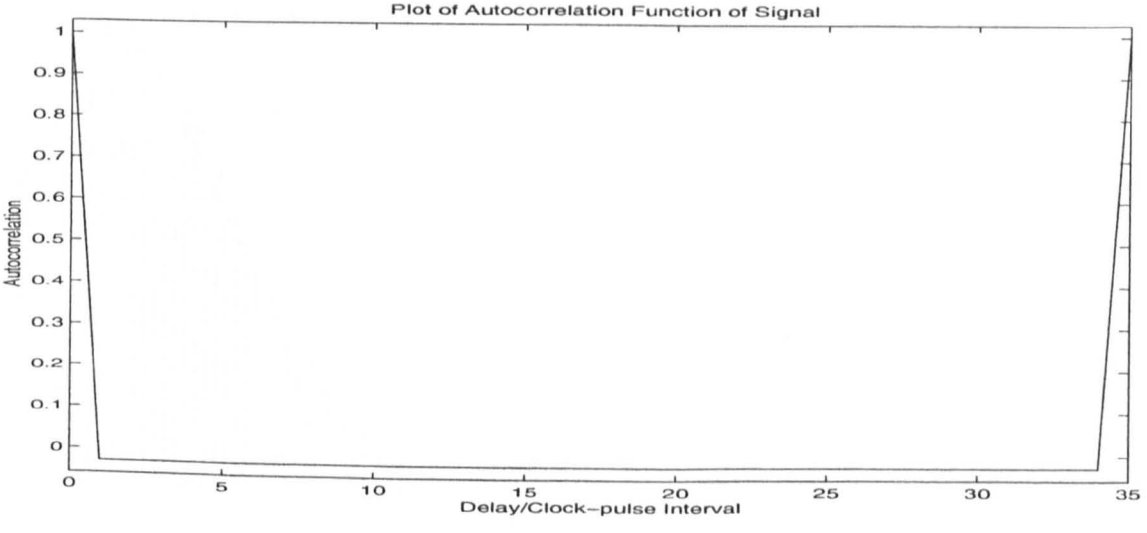

Figure 2.4. Autocorrelation function of a TPB signal of length 35.

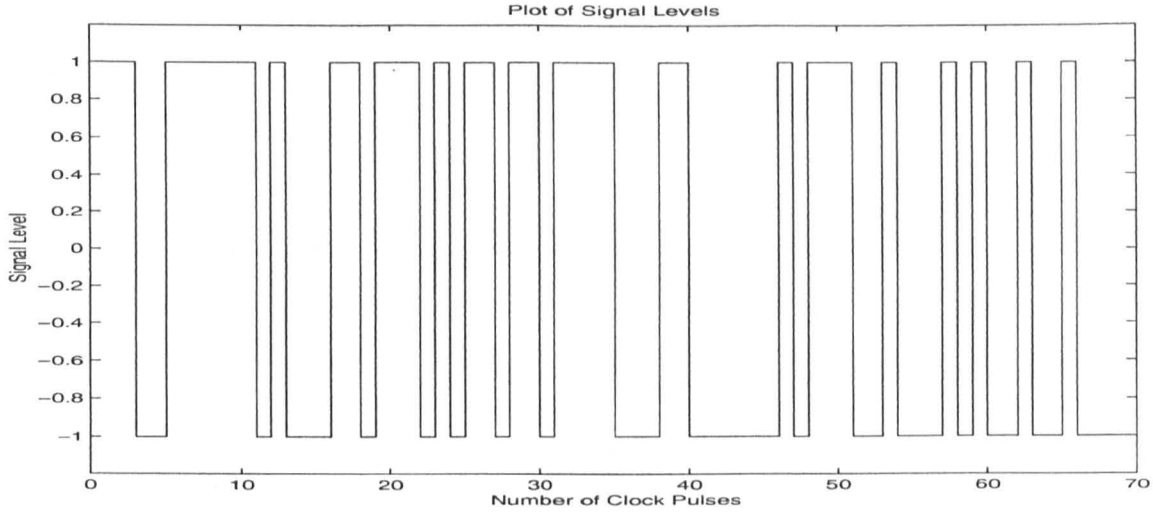

Figure 2.5. Inverse-repeat TPB signal of length 70.

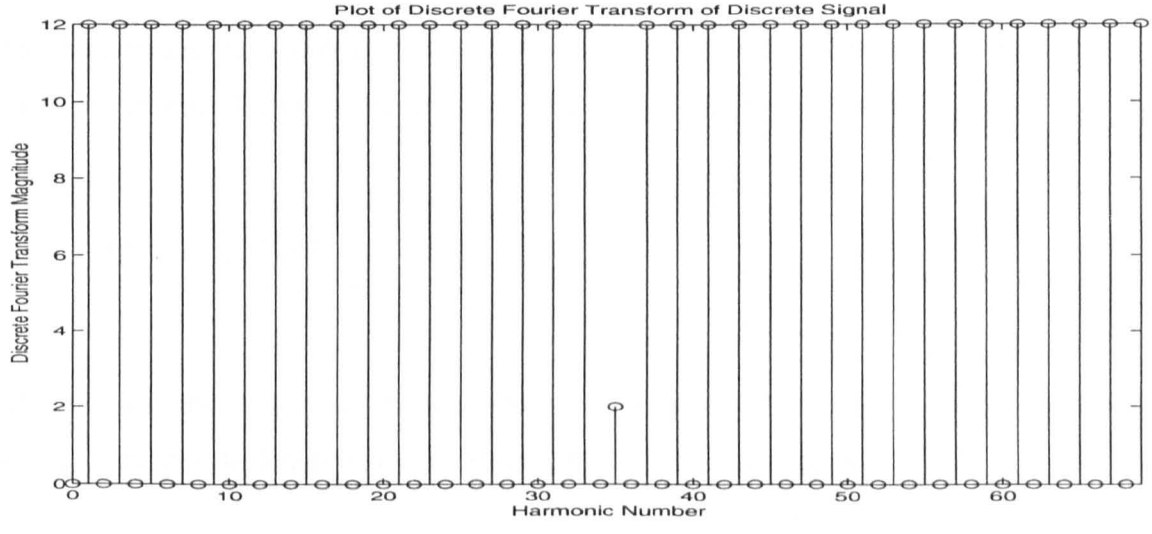

Figure 2.6. Discrete Fourier transform of an inverse-repeat TPB signal of length 70.

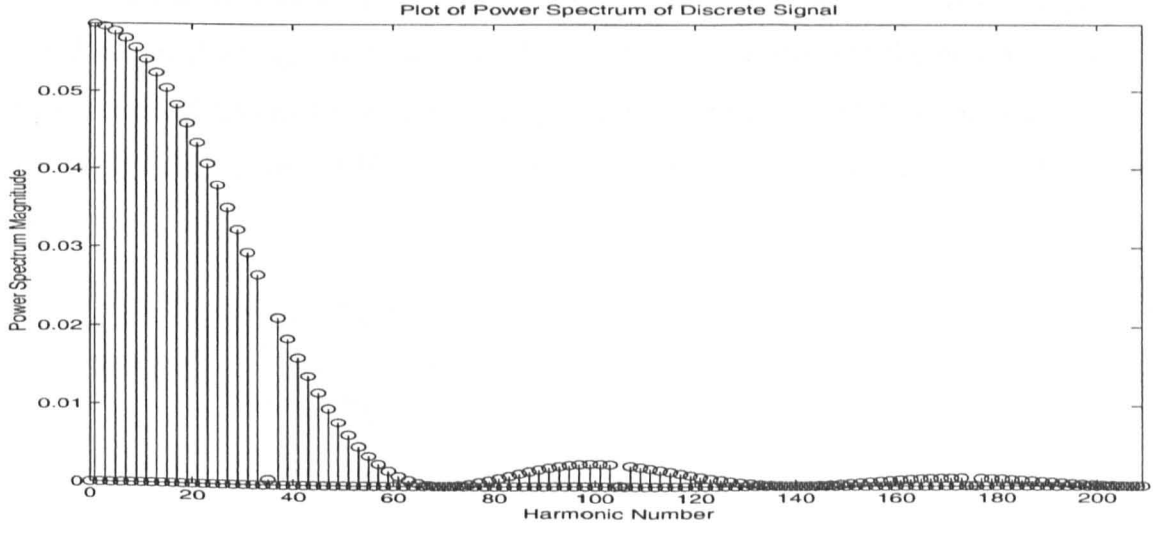

Figure 2.7. Power spectrum of an inverse-repeat TPB signal of length 70.

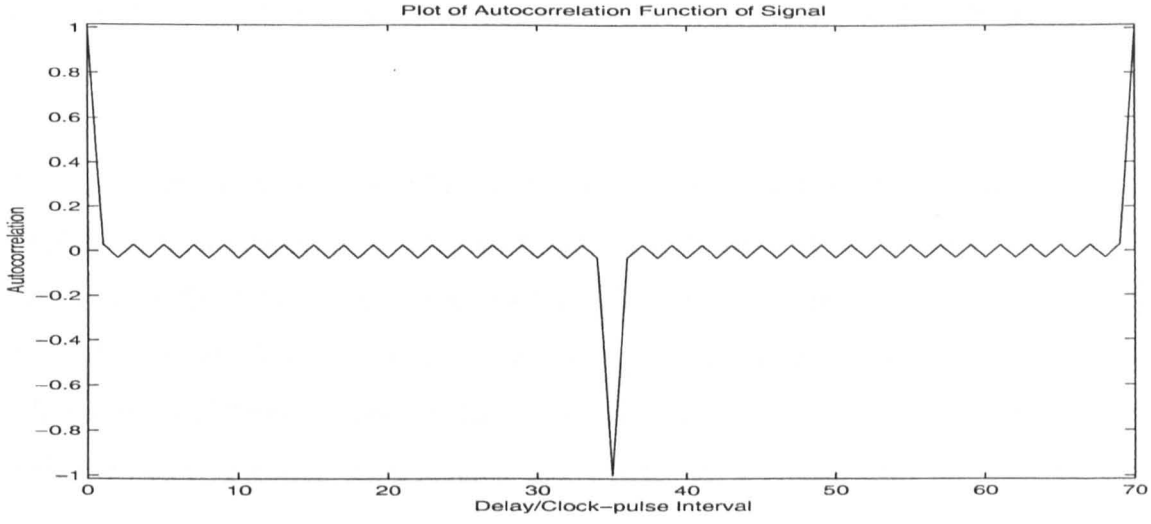

Figure 2.8. Autocorrelation function of an inverse-repeat TPB signal of length 70.

The routine also incorporates functions to calculate different measures of signal quality. The Performance Index for Perturbation Signals (PIPS) [4] is given by

$$\text{PIPS} = \frac{200\sqrt{u_{\text{rms}}{}^2 - u_{\text{mean}}{}^2}}{u_{\max} - u_{\min}}\% \tag{2.9}$$

where u_{rms} is the rms value of the signal, u_{mean} is the mean value of the signal, and $u_{\max}$ and $u_{\min}$ are the maximum and minimum values of the signal respectively.

It is used to evaluate the performance of a signal in maximising the power in specified harmonics within amplitude constraints. This is important since the signal amplitude should be small enough to negate the effects of nonlinearities and the power contained in the specified harmonics should be large enough to negate the effects of noise. In the case of periodic signals, PIPS can be expressed in terms of the harmonic content of u, so that

$$\text{PIPS} = \frac{200\sqrt{\sum_{k=1}^{N-1}|U(k)|^2}}{N(u_{\max} - u_{\min})}\% \tag{2.10}$$

where $U(k)$ is the discrete Fourier transform (DFT) magnitude of the kth harmonic of $u(t)$.

An important advantage of PIPS is that it is independent of the signal mean and amplitude, thus making it more useful and versatile than other measures such as the Peak Factor, Crest Factor and Form Factor, which were originally used to quantify sine wave distortions. These measures have an additional disadvantage of being defined differently by different authors. A further advantage of PIPS is that it is 0% for a signal with the worst possible performance and 100% for a signal with the best possible performance. A pseudo-random binary signal has a PIPS of $100\sqrt{1-\frac{1}{N^2}}$ % while a QRT signal has a PIPS of $100\sqrt{1-\frac{1}{N}}$ % [4]. These tend to 100% as N increases. It should be noted that a binary signal in which the two levels have an equal duration or number of occurrences has a PIPS of 100% [4] and this is a great advantage over multi-level signals.

PIPS can be modified to suit particular applications and the derivation thus obtained is designated PIPS(Effective) or PIPSE [4], which is given by

$$\text{PIPSE} = \frac{200\sqrt{\sum_{k=1}^{R}|C'_u(k)|^2}}{(u_{\max}-u_{\min})}\% \qquad R < N/2 \qquad (2.11)$$

where $|C'_u(k)|^2$ is the power of the kth harmonic of $u(t)$ and the first R nonzero harmonics are specified for the identification. For a pseudo-random binary signal, PIPSE is $100\sqrt{\frac{2(N+1)}{N^2}\sum_{k=1}^{R}\left(\frac{\sin \pi k / N}{\pi k / N}\right)^2}$ % whereas for a QRT signal, PIPSE is $100\sqrt{\frac{2}{N}\sum_{k=1}^{R}\left(\frac{\sin \pi k / N}{\pi k / N}\right)^2}$ % [4].

The use of PIPS and PIPSE alone as a measure of signal quality would not take into account the possibility that the power contained in one of the specified harmonics is low, and hence result in an inaccurate estimation of the frequency response at that particular frequency. A frequency domain measure of signal quality is then used which is the effective minimum ratio between the actual harmonic amplitude and the specified harmonic amplitude at any of the specified harmonics, EMINE [4]. This is defined as

$$\text{EMINE} = 100 \underset{k=1,2,\ldots,R}{\text{minimum}} \frac{|C'_u(k)|}{\sqrt{\frac{1}{R}\sum_{k=1}^{R}|C'_u(k)|^2}} \% \qquad R < N/2 \tag{2.12}$$

EMINE is also independent of the signal mean and amplitude, and has a value of 0% for a signal with the worst possible performance and 100% for a signal with the best possible performance. For both pseudo-random binary signals and QRT signals, the Rth harmonic has the least power, so that EMINE is $100 \dfrac{\left(\dfrac{\sin \pi R / N}{\pi R / N}\right)}{\sqrt{\dfrac{1}{R}\sum_{k=1}^{R}\left(\dfrac{\sin \pi k / N}{\pi k / N}\right)^2}}$ % [4].

The values of PIPSE and EMINE for a specification with R consecutive harmonics are given in Table 2.1. From the table, it can be seen that as R increases, PIPSE increases since more signal power is now at the specified harmonics. However, this is accompanied by a decrease in EMINE. It is worth noting that PIPSE for a QRT signal is only slightly lower than that of a pseudo-random binary signal, since there is only one element at the signal level zero in a period.

Signal type	R	PIPSE (%)	EMINE (%)
PRB	15	48.4	98.6
PRB	30	66.9	94.0
PRB	63	87.9	73.0
QRT	15	48.2	98.6
QRT	30	66.6	94.0
QRT	63	87.5	73.0

Table 2.1. PIPSE and EMINE for different values of R using a pseudo-random binary (PRB) signal and a QRT signal, both with $N = 127$.

2.3.4 List of Functions

The MATLAB routine has a script file and 29 function files. The routine is started by running the script file and it will call other function files as necessary. A brief description of the purpose of the script file and all the function files is given below. The script file, *prs_perturbation*, is listed first followed by the function files in alphabetical order. A complete program is given in the Appendix.

Script file :

prs_perturbation : To generate binary and ternary pseudo-random signals. These signals can have either all harmonics present or only odd harmonics present.

Function files :

autocorrelation : To calculate the periodic autocorrelation of a signal.

autocorrelation_plot : To display the autocorrelation function graphically (and numerically if required).

calculate_EMINE : To calculate EMINE as given in equation (2.12).

calculate_PIPS : To calculate PIPS as given in equation (2.10).

calculate_PIPSE : To calculate PIPSE as given in equation (2.11).

calculate_R0 : To calculate the limit in which unsystematic errors due to second order nonlinearities cannot be eliminated when an MLB signal is used to estimate the system weighting function (see Section 2.3.1.3).

calculate_R1 : To calculate the limit in which unsystematic errors due to third order nonlinearities cannot be eliminated when an MLB signal is used to estimate the system weighting function (see Section 2.3.1.3).

DFT_plot : To calculate the discrete Fourier transform magnitude and present it graphically.

HAB : To generate HAB signals.

HAB_exist : To check whether a HAB signal is present for a particular length.

inverse_repeat : To generate an inverse-repeat signal from a non inverse-repeat signal.

MLB : To generate MLB signals. The characteristic equation can be specified by the user. Otherwise, the function *mlbs* in the MATLAB Frequency Domain System Identification Toolbox is used to generate a 'default' MLB signal.

MLB_exist : To check whether an MLB signal is present for a particular length.

poly_check : To check if the characteristic equation of a shift register is irreducible and primitive. In other words, this checks if the signal generated by the shift register will be a maximum length signal (an MLB signal in this case). (This was discussed in Section 2.3.1.1.)

poly_MLB : To generate an MLB signal when a characteristic equation is specified.

power_plot : To calculate the power spectrum and display the results graphically (and numerically if required).

prime : To check if a particular integer is a prime.

prs_all_harmonics	:	To generate binary and ternary pseudo-random signals with all harmonics present.
prs_odd_harmonics	:	To generate binary and ternary pseudo-random signals with only odd harmonics present.
QRB	:	To generate QRB signals.
QRB_exist	:	To check whether a QRB signal is present for a particular length.
QRT	:	To generate QRT signals.
QRT_exist	:	To check whether a QRT signal is present for a particular length.
second_shift	:	To calculate the shift in the resulting signal when an MLB signal is shifted by a particular delay and then added on to the original (non shifted) signal.
sequence_plot	:	To plot the generated signal.
signal_display	:	To display the generated signal, its discrete Fourier transform, power spectrum and autocorrelation function by calling the functions *sequence_plot*, *DFT_plot*, *power_plot* and *autocorrelation_plot* respectively. It also allows the user to save the data relating to the generated signal in an external file.
third_shift	:	To calculate the shift in the resulting signal when two MLB signals which are shifted by two different values of delay are added on to the original (non shifted) signal.
TPB	:	To generate TPB signals.
TPB_exist	:	To check whether a TPB signal is present for a particular signal.

A routine to find the least primitive root for any given prime number, *primitive_root*, was also written but is not part of the new MATLAB routine. However, the results generated using *primitive_root* are incorporated into the MATLAB routine within the function *HAB*.

2.4 Extension to Multi-level Pseudo-random Signals

2.4.1 Introduction to Nonlinear System Identification

The identification of nonlinear structures such as the Wiener and Hammerstein models using pseudo-random signals is investigated in [18]. It is shown that for the Wiener model in which a dynamic linear subsystem precedes an instantaneous nonlinearity, the performance of the signal measured using PIPS is the same as for a linear system. It is concluded in [18] that a binary signal with a PIPS of 100% will give the best performance with the Wiener model.

More complicated analysis is needed for the Hammerstein model, in which a dynamic linear subsystem succeeds an instantaneous nonlinearity. Such a structure with quadratic nonlinearity is considered in [18]. In this case, a perturbation signal with at least three levels is needed to identify the linear and nonlinear parts of the system. This is because a binary signal will produce a constant at the output of the quadratic nonlinearity. It is also shown in [18] that it is desirable for the signal to have even harmonics suppressed and its odd harmonics uniform. In addition, the square of the signal should have its odd harmonics suppressed and its even harmonics, apart from the zero harmonic, uniform. This is because the square of the signal will be an input to a linear subsystem, and hence the performance criteria should be applied to it as well. The theory can be extended to Hammerstein models with a higher order of nonlinearity. To identify a system with N_D-1 order of nonlinearity, the signal must have at least N_D levels [32]. The optimum signal levels were found in [32] by minimising the condition number of the associated Vandermonde matrix [33].

Some properties of multi-level (three-level, four-level and five-level) signals will be considered in the next section. It is worth noting that these signals (and those with a higher number of levels) can be generated from prime and extension fields [34] using the computer software GALOIS [8]. In Section 2.4.3, the optimum signal design for the identification of Hammerstein models will be investigated.

2.4.2 Properties of Multi-level Signals

Multi-level signals $u(i)$ may be obtained from maximum length sequences $s(i)$ generated in the Galois field GF(q). Such a sequence has a period $N = q^n-1$ and can be divided into q-1 subperiods. Each period of the signal contains q^{n-1} occurrences of each elements, except element 0 which occurs only q^{n-1}-1 times.

2.4.2.1 Three-level Signals

Several properties of three-level maximum length sequences generated from GF(3) are given in [35, 36]. Examples of shift register feedback connections resulting in such a signal for $2 \leq n \leq 7$ are given in [35] and these were derived from a list of primitive polynomials tabulated in [37]. Since the signal can be divided into two subperiods, it is therefore an inverse-repeat signal.

For such a signal $u(i)$, the magnitudes of the harmonics are given in [18] by

$$\begin{aligned} |U(k)| &= 2 \times 3^{(n-1)/2} && k \text{ odd} \\ |U(k)| &= 0 && k \text{ even} \end{aligned} \tag{2.13}$$

PIPS is $100\sqrt{2(N+1)/3N}\%$, which tends to 81.6% as N increases [18]. In the identification of Hammerstein models with N_D - 1 = 2, the magnitudes of the harmonics of the square of the signal $v(i)$ are of importance as the signal $v(i) = u(i)^2$ will be an input to the linear subsystem. The magnitudes of these harmonics are given by

$$\begin{aligned} |V(k)| &= 2 \times 3^{n-1} && k = rN \\ |V(k)| &= 2 \times 3^{(n-2)/2} && k \text{ even} \neq rN \\ |V(k)| &= 0 && k \text{ odd} \end{aligned} \tag{2.14}$$

and PIPS is $100\sqrt{2(N+1)(N-2)/9N^2}\%$, which tends to 47.1% as N increases [18]. Thus, both the signal and the square of the signal possess the required property for the identification of the Hammerstein model (with the nonlinear part being confined to a quadratic nonlinearity).

2.4.2.2 Four-level Signals

Four-level maximum length sequences based on GF(4) are considered in [38, 39]. If the elements (0, 1, 2, 3) of the sequence are mapped into signal levels (u_1, u_2, u_3, u_4), the mean value M is given in [39] by

$$M = \frac{u_1 + u_2 + u_3 + u_4}{4} + \frac{u_2 + u_3 + u_4 - 3u_1}{4(4^m - 1)} \tag{2.15}$$

The normalised autocorrelation function is given by

$$R(0) = \frac{u_1^2 + u_2^2 + u_3^2 + u_4^2}{4} + \frac{u_2^2 + u_3^2 + u_4^2 - 3u_1^2}{4(4^m - 1)} \tag{2.16}$$

$$R\left(\frac{N}{3}\right) = R\left(\frac{2N}{3}\right) = \frac{u_1^2 + u_2u_3 + u_3u_4 + u_4u_2}{4} + \frac{u_2u_3 + u_3u_4 + u_4u_2 - 3u_1^2}{4(4^m - 1)} \tag{2.17}$$

$$R(k) = \frac{(u_1 + u_2 + u_3 + u_4)^2}{16} + \frac{(u_1 + u_2 + u_3 + u_4)^2 - 16u_1^2}{16(4^m - 1)} \tag{2.18}$$

where k is not an integer multiple of $N/3$.

If we consider equal spacing between the signal levels by taking $u_1 = C$, $u_2 = C + A$, $u_3 = C + 2A$, $u_4 = C + 3A$ as was done in [39], equations (2.17) and (2.18) can be simplified to

$$R\left(\frac{N}{3}\right) = R\left(\frac{2N}{3}\right) = C^2 + \frac{12AC + 11A^2}{4}\left(\frac{4^m}{4^m - 1}\right) \tag{2.19}$$

$$R(k) = C^2 + \frac{12AC + 9A^2}{4}\left(\frac{4^m}{4^m - 1}\right) \tag{2.20}$$

Since equation (2.19) cannot be made equal to equation (2.20) (unless $A = 0$, resulting in a constant signal), the autocorrelation function is not primitive and the harmonics are not uniform in magnitude. Also, inverting every other member of the signal does not produce an inverse-repeat signal with uniform odd harmonics.

In order to obtain a four-level signal with uniform harmonics (except at dc), the mapping levels must be made asymmetrical. Unfortunately, this results in a signal with

the separation of levels very far from being equal, thus greatly limiting its usefulness in a perturbation test. The equality of spacing between the levels can be measured using its FIT value. For a signal with equal spacing, the signal levels are

$$\left(u_1, u_1 + \frac{u_4 - u_1}{3}, u_4 - \frac{u_4 - u_1}{3}, u_4\right) \tag{2.21}$$

For a particular four-level signal, the error term is defined as

$$\text{error} = \left|u_2 - \left(u_1 + \frac{u_4 - u_1}{3}\right)\right| + \left|u_3 - \left(u_4 - \frac{u_4 - u_1}{3}\right)\right| \tag{2.22}$$

and FIT is then defined as

$$\text{FIT} = 100\left(\frac{1 - \text{error}}{u_4 - u_1}\right)\% \tag{2.23}$$

Some mapping levels for four-level signals with uniform harmonics (except at dc) are given in Table 2.2, together with their values of PIPS (defined in (2.10)) and FIT.

u_1	u_2	u_3	u_4	PIPS (%)	FIT (%)
-8	-7	1	8	81.3	62.5
-9	-7	-1	9	77.8	55.6
-18	-17	7	18	86.1	66.7
-25	-23	7	25	84.0	68.0
-25	-17	-7	25	76.0	52.0
-32	-31	17	32	89.1	58.3
-32	-23	-7	32	76.6	53.1
-49	-47	23	49	87.8	61.9
-49	-41	1	49	79.6	59.2
-49	-31	-17	49	75.5	51.0
-50	-49	31	50	91.0	53.3
-50	-41	-1	50	79.0	58.0
-81	-79	47	81	90.1	55.6
-81	-73	17	81	82.7	65.4
-81	-49	-31	81	75.3	50.6

Table 2.2. PIPS and FIT values for several four-level signals with uniform harmonics.

2.4.2.3 Five-level Signals

For pseudo-random sequences $u(i)$ obtained from GF(5), the mapping of the sequence elements into signal levels which result in uniform harmonics can be written as

$$\begin{aligned} u_1 &= 0 \\ u_2 &= -u_5 = a \\ u_3 &= -u_4 = b \end{aligned} \tag{2.24}$$

When $a > 1$ and $b = 1$, $u(i)$ has five levels. Typically, $a = 2$ since this gives equal spacing between the levels. The magnitude of the harmonics are then given by

$$\begin{aligned} |U(k)| &= 4.47 \times 5^{(n-1)/2} && k \text{ odd} \\ |U(k)| &= 0 && k \text{ even} \end{aligned} \tag{2.25}$$

The closer a is to unity, the larger the value of PIPS.

The field elements can also be mapped such that the resulting signal is three-level by making $a = 0$ and $b = 1$, or $a = b = 1$. In the former case,

$$\begin{aligned} |U(k)| &= 2 \times 5^{(n-1)/2} && k \text{ odd} \\ |U(k)| &= 0 && k \text{ even} \end{aligned} \tag{2.26}$$

In the latter case

$$\begin{aligned} |U(k)| &= 2.83 \times 5^{(n-1)/2} && k \text{ odd} \\ |U(k)| &= 0 && k \text{ even} \end{aligned} \tag{2.27}$$

However, the square of the signal do not possess uniform even harmonics, unlike the three-level signals generated from GF(3).

2.4.3 Design of Multi-level Signals for the Identification of Hammerstein Models

For the identification of a nonlinear system with the Hammerstein structure, a multi-level perturbation signal u is required. The signal must have at least N_D different levels if the order of the nonlinearity is N_D - 1. For the present, assume that every level has the same number of occurrences. Without loss of generality it may also be assumed that $u_1 > u_2 > \ldots > u_{N_D}$, and that $u_1 = 1$ and $u_{N_D} = -1$. Despite the fact that intuition may suggest that the levels be equally spaced, this may not be the optimal solution.

To formulate the problem, the output of the nonlinear system y is expressed as a function of the input u.

$$y = a_1 u^{N_D-1} + a_2 u^{N_D-2} + \ldots + a_{N_D-1} u + a_{N_D} \tag{2.28}$$

Then the output of the nonlinearity at each signal level is given by

$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ y_{N_D-1} \\ y_{N_D} \end{bmatrix} = \begin{bmatrix} u_1^{N_D-1} & u_1^{N_D-2} & .. & u_1 & 1 \\ u_2^{N_D-1} & u_2^{N_D-2} & .. & u_2 & 1 \\ . & . & .. & . & . \\ u_{N_D-1}^{N_D-1} & u_{N_D-1}^{N_D-2} & .. & u_{N_D-1} & 1 \\ u_{N_D}^{N_D-1} & u_{N_D}^{N_D-2} & .. & u_{N_D} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ a_{N_D-1} \\ a_{N_D} \end{bmatrix} \tag{2.29}$$

where $\begin{bmatrix} u_1^{N_D-1} & u_1^{N_D-2} & .. & u_1 & 1 \\ u_2^{N_D-1} & u_2^{N_D-2} & .. & u_2 & 1 \\ . & . & .. & . & . \\ u_{N_D-1}^{N_D-1} & u_{N_D-1}^{N_D-2} & .. & u_{N_D-1} & 1 \\ u_{N_D}^{N_D-1} & u_{N_D}^{N_D-2} & .. & u_{N_D} & 1 \end{bmatrix}$ is the Vandermonde matrix of $\begin{bmatrix} u_1 \\ u_2 \\ . \\ u_{N_D-1} \\ u_{N_D} \end{bmatrix}$.

Equation (2.29) may be written as

$$y = V_{N_D}(u)a \tag{2.30}$$

hence

$$a = \left(V_{N_D}(u)\right)^{-1} y \tag{2.31}$$

There are several ways to select the optimal values of u. These are given below :

(a) In the context of a measurement problem, errors occur in the measurements y, and the requirement is to minimise the sensitivity of the solution to these errors. The sensitivity matrix is

$$\frac{da}{dy} = \left(V_{N_D}(u)\right)^{-1} \tag{2.32}$$

and it is therefore required to minimise the sum of all sensitivities

$$\sum_i \sum_j \left| \left(V_{N_D}(u)\right)^{-1}_{i,j} \right| \tag{2.33}$$

(b) It is also possible to minimise the greatest sensitivity given by

$$\max_i \left(\max_j \left(\left| \left(V_{N_D}(u)\right)^{-1}_{i,j} \right| \right) \right) \tag{2.34}$$

(c) In the context of a least-squares polynomial fitting problem, errors occur in y and the effect of these errors is minimised by minimising

$$\left\| \left(V_{N_D}(u)\right)^{-1} \right\| \tag{2.35}$$

(d) The problem can also be treated as a matrix inversion problem where numerical errors occur in the inversion of $V_{N_D}(u)$. In this case, the quantity to minimise is the 2-norm of the condition number of $V_{N_D}(u)$

$$\mathrm{cond}_2\left(V_{N_D}(u)\right) \tag{2.36}$$

as was done in [32]. This is defined as

$$\mathrm{cond}_2 V_{N_D} = \left[\frac{\lambda_{\max}(V_{N_D}{}^T V_{N_D})}{\lambda_{\min}(V_{N_D}{}^T V_{N_D})} \right]^{\frac{1}{2}} \tag{2.37}$$

where λ represents the singular values of a matrix.

These four measures all give the same result when $N_D = 3$. In this case

$$V_{N_D}(u) = \begin{bmatrix} u_1^2 & u_1 & 1 \\ u_2^2 & u_2 & 1 \\ u_3^2 & u_3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ u_2^2 & u_2 & 1 \\ 1 & -1 & 1 \end{bmatrix} \tag{2.38}$$

and

$$\left(V_{N_D}(u)\right)^{-1} = \begin{bmatrix} \dfrac{1}{(u_1-u_2)(u_1-u_3)} & \dfrac{1}{(u_2-u_3)(u_2-u_1)} & \dfrac{1}{(u_3-u_1)(u_3-u_2)} \\ -\dfrac{u_2+u_3}{(u_1-u_2)(u_1-u_3)} & -\dfrac{u_3+u_1}{(u_2-u_3)(u_2-u_1)} & -\dfrac{u_1+u_2}{(u_3-u_1)(u_3-u_2)} \\ \dfrac{u_2u_3}{(u_1-u_2)(u_1-u_3)} & \dfrac{u_3u_1}{(u_2-u_3)(u_2-u_1)} & \dfrac{u_1u_2}{(u_3-u_1)(u_3-u_2)} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{1}{2(1-u_2)} & -\dfrac{1}{1-u_2^2} & \dfrac{1}{2(1+u_2)} \\ \dfrac{1}{2} & 0 & -\dfrac{1}{2} \\ -\dfrac{u_2}{2(1-u_2)} & \dfrac{1}{1-u_2^2} & \dfrac{u_2}{2(1+u_2)} \end{bmatrix} \tag{2.39}$$

In this case all four requirements are satisfied when $u_2 = 0$ which results in

$$\left(V_{N_D}(u)\right)^{-1} = \begin{bmatrix} 0.5 & -1 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.40}$$

The four measures do not, however, give the same results when $N_D > 3$. These are tabulated in Table 2.3 for the cases when $N_D = 4$ and 5. Methods (a) to (d) correspond to minimising the quantities given in (2.33) to (2.36) respectively.

N_D	Method	u_2	u_3	u_4
4	a	0.425	-0.425	
4	b	0.575	-0.575	
4	c	0.545	-0.545	
4	d	0.520	-0.520	
5	a	0.655	0	-0.655
5	b	0.840	0	-0.840
5	c	0.745	0	-0.745
5	d	0.725	0	-0.725

Table 2.3. Optimal signal levels using different methods.

The results for $N_D = 3$, 4, and 5 all have a symmetry of the form

$$u_{N_D-r+1} = -u_r \qquad r = 1, 2, \ldots \tag{2.41}$$

Also, as the only known four-level signals with ideal harmonics are very asymmetric, the above results indicate that they are not likely to be suitable for this application.

The above analysis needs to be modified if a nonlinearity of order N_D - r (where $r \neq 1$) needs to be identified instead, and if some or all of the signal levels occur more than once. In the first case, the number of columns of the Vandermonde matrix is reduced, while in the second case, the number of rows of the matrix is increased. This modification does not change the basic solution of the problem except that the measures given in methods (a) to (d) are applied to a modified version of the Vandermonde matrix *Vm*. Due to the fact that the modified matrix is no longer a square matrix, it is now required to minimise

(a) $$\sum_i \sum_j \left| \left(\left(Vm(u)^T Vm(u) \right)^{-1} Vm(u)^T \right)_{i,j} \right| \tag{2.42}$$

(b) $$\max_i \left(\max_j \left(\left| \left(\left(Vm(u)^T Vm(u) \right)^{-1} Vm(u)^T \right)_{i,j} \right| \right) \right) \tag{2.43}$$

(c) $$\left\| \left(Vm(u)^T Vm(u) \right)^{-1} Vm(u)^T \right\| \tag{2.44}$$

(d) $$\mathrm{cond}_2 \left(Vm(u)^T Vm(u) \right) \tag{2.45}$$

When either a four-level or a five-level signal is used to identify a quadratic nonlinearity, all four requirements are satisfied by the same degenerate case for which $u_2 = u_3 = u_4 = 0$. Results obtained when a five-level signal is used to identify a cubic nonlinearity are given in Table 2.4. Here, methods (a) to (d) correspond to minimising (2.42) to (2.45) respectively.

Method	a	b	c	d
$u_2 = - u_4$	0.467	0.577	0.543	0.526

Table 2.4. Optimal signal levels for a five-level signal used to identify a cubic nonlinearity. $u_1 = - u_5 = 1$, $u_3 = 0$.

If each signal level occurs p times instead of once, it might be expected that the four measures involved would be invariant, or would decrease as p increases since more data is obtained with an increased signal length. It was found that the measures given in (2.42) and (2.45) are invariant, while the measure given in (2.43) decreases at a rate proportional to p, and that in (2.44) decreases at a rate proportional to $\sqrt{p}$.

For the purpose of comparing the performance of different signal levels, it is desirable that the measures involved are not dependent on the number of times the levels occur, provided that the number of occurrences of each level is approximately the same. It is therefore convenient to adopt the four invariant measures.

(a) $$\sum_i \sum_j \left| \left(\left(Vm(u)^T Vm(u) \right)^{-1} Vm(u)^T \right)_{i,j} \right| \tag{2.46}$$

(b) $$p * \max_i \left(\max_j \left(\left| \left(\left(Vm(u)^T Vm(u) \right)^{-1} Vm(u)^T \right)_{i,j} \right| \right) \right) \tag{2.47}$$

(c) $$\sqrt{p} * \left\| \left(Vm(u)^T Vm(u) \right)^{-1} Vm(u)^T \right\| \tag{2.48}$$

(d) $$\mathrm{cond}_2 \left(Vm(u)^T Vm(u) \right) \tag{2.49}$$

These four measures can also be used for multi-level pseudo-random signals in which the number of occurrences of all nonzero levels is equal and that for the zero level is one less. The invariant measures in (2.46) to (2.49) for a three-level signal generated from GF(3) with levels +1, 0 and -1 are given in Table 2.5.

Length, N	8	26	80	242	728	$3p$
No. of ± 1, p	3	9	27	81	243	p
No. of 0	2	8	26	80	242	p
Measure (a)	4.00	4.00	4.00	4.00	4.00	4.00
Measure (b)	1.50	1.13	1.04	1.01	1.00	1.00
Measure (c)	1.81	1.59	1.54	1.52	1.51	1.51
Measure (d)	14.26	11.36	10.70	10.50	10.44	10.40

Table 2.5. Invariant measures for a three-level signal.

Since the measures given in Table 2.5 are relatively invariant, they are proven to be very useful in evaluating the signal performance. In fact, the slight variations seen are due to the fact that the number of occurrences of the signal levels is not exactly equal (except for the last column of Table 2.5). Indeed, the values of p are not exact, due to the above reasons. As the signal length increases, the number of occurrences of the signal levels becomes more like equal and the signal measures tend towards that given in the last column of Table 2.5.

It is of interest to further investigate the signal measures if the number of occurrences of the signal levels are significantly different. Unfortunately, when this is the case, the measures given in (2.46) to (2.49) cannot be used because it is no longer possible to specify a value for p. The measures in (2.42) to (2.45) have to be used instead. Examples are given in Table 2.6 for a signal of length $N = 80$ generated from GF(9) and mapped into the three levels +1, 0 and -1.

No. of ±1	9	18	27	36
No. of 0	62	44	26	8
Measure (a)	4.00	4.00	4.00	4.00
Measure (b)	0.06	0.03	0.04	0.13
Measure (c)	0.28	0.25	0.30	0.51
Measure (d)	6.45	6.34	10.70	38.08
PIPS (%)	47.4	67.1	82.2	94.9

Table 2.6. Measures for a three-level signal generated from GF(9) with signal levels +1, 0 and -1.

It can be seen from Table 2.6 that the signal measures (b) to (d) are the worst (largest) when the number of occurrences of signal levels ±1 increases. Hence, the requirement to most accurately perform the matrix computation in the presence of noise contradicts with that to maximise the signal power within a specified peak-to-peak amplitude (PIPS). In view of this, the signal power in Table 2.6 was normalised to a root-mean-

square value of unity, and the signal measures were recomputed as in Table 2.7. It can be seen that despite normalising the signal power, the above observation still holds.

No. of $\pm V$	9	18	27	36
No. of 0	62	44	26	8
Measure (a)	1.90	2.68	3.28	3.81
Measure (b)	0.03	0.02	0.03	0.12
Measure (c)	0.13	0.17	0.24	0.48
Measure (d)	6.45	6.34	10.70	38.08
V	2.11	1.49	1.22	1.05

Table 2.7. Measures for a three-level signal generated from GF(9) with an rms power of unity.

2.5 Conclusions

MLB signals are the best known class of pseudo-random signals due to their ease of generation using feedback shift registers. The characteristic equation in the delays of the shift register must be irreducible and primitive in order to result in an MLB signal. This class of signal possesses many unique properties such as the shift-and-add property which is useful in some applications for example, the identification (detection) of processes with direction-dependent dynamics (see Chapter 3). The effect of nonlinearities on the measurement of weighting functions by input-output crosscorrelation is also a factor which needs to be taken into consideration when making a choice of an MLB signal of a particular length. Furthermore, the use of an inverse-repeat signal eliminates the effect of even order nonlinearities and this is very useful in the estimation of linear dynamics for processes with nonlinearities. The MATLAB routine incorporates functions to generate MLB signal for any specified characteristic equation (provided the characteristic equation is irreducible and primitive), and investigate the effects of the shift-and-add property and the errors on the estimate of the weighting function caused by nonlinearities in the system.

Other classes of binary and near-binary signal such as the QRB, HAB and TPB signals have identical autocorrelation properties to that of an MLB signal while the QRT signal has almost identical autocorrelation properties. These signals are useful in filling in the gaps in between successive periods of the MLB signal. The design of these classes of signals was considered.

The MATLAB routine allows pseudo-random signals with all harmonics present and those with only odd harmonics present (inverse-repeat) to be generated. Other useful functions include graphical representations of the signal, its discrete Fourier transform magnitude, power spectrum and autocorrelation, and functions to calculate different measures of signal quality such as PIPS, PIPSE and EMINE. The data generated can be directly written to an external file. Thus, the MATLAB routine is a useful and convenient tool for generating pseudo-random signals for the purpose of system identification. The signals generated are applied to the detection of the departure from linearity for processes with direction-dependent dynamics (in Chapter 3), the estimation of linear dynamics for such processes (in Chapter 4), and the modelling of such processes using the Wiener model (in Chapter 5) and neural network model (in Chapter 6).

For identification of the Wiener structure, a binary signal with a PIPS of 100% is the best signal to use, as is the case for linear systems. However, the above is not true for the identification of Hammerstein models and the use of multi-level maximum length signals for this purpose was investigated. While three-level and five-level signals possess some desirable properties, it was found that four-level signals have a limited usefulness since they cannot be made inverse-repeat.

Three new signal measures were suggested, alongside with another measure already proposed in [32], to evaluate the quality of the Vandermonde matrix in the case of identifying a Hammerstein model. These four measures are the sum of all sensitivities in solving for the coefficients of the nonlinearity in the presence of noise, its greatest sensitivity, the norm of the inverse Vandermonde matrix, and the 2-norm condition number of the Vandermonde matrix. The Vandermonde matrix can also be modified if the order of the nonlinearity is less than the number of signal levels minus one, and if

some or all of the levels occur more than once. The four measures above were modified accordingly and invariant measures were suggested which are independent of the signal length. It was found that the requirement which results in the best performance measured by these contradicts with that to maximise the power within constraints of the signal amplitude (that is, to maximise PIPS (or PIPSE)).

Chapter 3

Detection of Departure from Linearity for Processes with Direction-dependent Dynamics

3.1 Abstract

Pseudo-random signals and multisines are applied to the detection of the departure from linearity for processes with direction-dependent dynamics. These processes have dynamics which depend on whether the output is increasing or decreasing. Analytical expressions for the process output are given for a first order process perturbed with a binary pseudo-random signal. Simulation results are presented for both first and second order processes when binary and multi-level pseudo-random signals, and multisine signals are applied. Identification (detection of the direction-dependent behaviour) is carried out in both time and frequency domains.

3.2 Introduction

A system with direction-dependent dynamics has dynamics which are different, according to whether the slope of the output is positive or negative. In this chapter, such a system which is linear in both directions is considered. Examples in the industry include gas-turbine engines (in which the input transducer has direction-dependent dynamic responses), chemical processes and nuclear reactors [26], steam-raising plants [25], distillation columns and polymerisation processes [40], and thermomechanical pulp refining [41, 42]. Such processes are also commonly found in the automotive sector where a damper may have a damping coefficient which is direction-dependent.

In order to detect the departure from linearity, one approach is to measure the responses to step function perturbations applied in the upward and downward directions. The problem with this is that, even with comparatively modest amounts of noise on the output, it becomes difficult to estimate the dynamics in each direction accurately. In this chapter, an alternative approach is described, which makes use of the noise rejection capabilities of input-output crosscorrelation. For a first order process perturbed with a binary signal, analytical expressions can be obtained for the process output. However, similar expressions could not be obtained for a first order process perturbed with a

multi-level signal and a second order process perturbed using a signal with any number of levels. This is because the sign of the slope of the output is not necessarily equal to the sign of the slope of the input (assuming that the input levels are symmetrical about zero) as in the case of a first order process perturbed with a binary signal.

Pseudo-random signals generated using the MATLAB routine (see Chapter 2) and those based on multi-level maximum length (MLML) sequences [36] are applied to detect the departure from linearity for processes with first and second order dynamics. Different classes of pseudo-random signals produce different patterns in the input-output crosscorrelation function and the existence of any coherent pattern can be explained using the shift-and-add and the shift-and-subtract properties. Random and Schroeder multisines [43] are also applied to the processes. Three different harmonic specifications are used. The first is the odd multisine with excitation lines at harmonic numbers f = 1, 3, 5, 7, 9, 11, etc. The second is the odd-odd multisine with excitation lines at f = 1, 5, 9, 13, etc. while the third has excitation lines at f = 1, 3, 9, 11, 17, 19, etc. The frequency spectrum of the output is analysed. The usefulness of the pseudo-random signals and multisines in detecting the presence of direction-dependent dynamics is discussed.

3.3 Process Simulation using MATLAB

MATLAB programs were written to simulate first and second order processes with direction-dependent dynamics. The programs allow the choice of input signal, the gains (which are set equal to unity in both directions for the rest of this chapter) and the dynamics in the positive and negative directions. For a first order process, the dynamics entered are the time constants for both directions of the output (T_U in equation (3.1A) and T_D in equation (3.1B)). The subscripts U and D denote the upward and downward directions of the output respectively.

$$Y(s) = \frac{1}{sT_U + 1} U(s) \qquad \text{sgn}(\dot{y}) > 0 \qquad (3.1A)$$

$$Y(s) = \frac{1}{sT_D + 1} U(s) \qquad \text{sgn}(\dot{y}) < 0 \qquad (3.1B)$$

where $U(s)$ and $Y(s)$ are the Laplace transforms of the input u and output y respectively.

For a second order process, the dynamics entered represent the time constants if their values are real. If complex values are entered, the real and imaginary parts are associated with the rate of exponential decay and an oscillatory response respectively. The dynamics entered into the program are T_{U1} and T_{U2} in equation (3.2A), and T_{D1} and T_{D2} in equation (3.2B). (For the rest of the thesis, T_{U1} and T_{U2} are represented as the two values of T_U, and T_{D1} and T_{D2} are represented as the two values of T_D.)

$$Y(s) = \frac{1}{(sT_{U1} + 1)(sT_{U2} + 1)} U(s) \qquad \text{sgn}(\dot{y}) > 0 \qquad (3.2A)$$

$$Y(s) = \frac{1}{(sT_{D1} + 1)(sT_{D2} + 1)} U(s) \qquad \text{sgn}(\dot{y}) < 0 \qquad (3.2B)$$

The main output of the programs is the output signal which is transferred directly into the MATLAB workspace.

The zero order hold assumption is incorporated into the program design. For a first order process, this has no effect on the output as the output changes immediately in response to a change in input. However, the above is not true for a second order process. The output is affected by the fact that the input is held constant between two consecutive clock-pulse intervals. This causes an additional zero in the overall transfer function of the system. The sampling interval was set to twenty times the clock-pulse interval for a second order process in order to achieve reasonable accuracy at the output.

The programs plot the graphs of the input and output signals, the input-output crosscorrelation function, the discrete Fourier transform of the input and output, and the gain and phase response of the system. These can also be saved in an associated file depending on the user's choice.

3.4 Detection of Direction-dependent Dynamics

3.4.1 Process with First Order Dynamics

3.4.1.1 Detection using Pseudo-random Binary Signals

For a first order process, the output y changes immediately in response to a change in the input u. y increases if u is larger than y when u is applied and decreases if otherwise. The value of y remains unchanged if u is equal to y when u is applied. For a binary signal with levels ± 1 and clock-pulse interval T (which is equal to the sampling interval in the first order case), it is shown in [25, 26] that

$$\mathrm{sgn}(\dot{y}) = \mathrm{sgn}(u) \tag{3.3A}$$

and

$$y_t = \kappa_t y_{t-1} + (1-\kappa_t)\, u_t \tag{3.3B}$$

where

$$\kappa_t = a + bu_t \tag{3.4A}$$

$$a = 0.5(\exp(-T/T_U) + \exp(-T/T_D)) \tag{3.4B}$$

$$b = 0.5(\exp(-T/T_U) - \exp(-T/T_D)) \tag{3.4C}$$

It is also shown in [25, 26] that the output consists of a constant, first order and higher order terms.

Constant term

$$Y_0 = K(1 + a + a^2 + \ldots) - \frac{1-a}{b} = -\frac{b}{1-a} \tag{3.5}$$

where

$$K = \frac{(1-a)^2}{b} - b \tag{3.6}$$

First order term

$$Y_1 = K\frac{b}{1-a}(u_t + au_{t-1} + a^2u_{t-2} + a^3u_{t-3} + a^4u_{t-4} + \ldots) \tag{3.7}$$

Second order terms

$$\begin{aligned} Y_2 = K\frac{b^2}{1-a}(&u_tu_{t-1} + au_{t-1}u_{t-2} + a^2u_{t-2}u_{t-3} + a^3u_{t-3}u_{t-4} + \ldots \\ &+ au_tu_{t-2} + a^2u_{t-1}u_{t-3} + a^3u_{t-2}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-3} + a^3u_{t-1}u_{t-4} + \ldots \\ &+ a^3u_tu_{t-4} + \ldots \\ &+ \ldots) \end{aligned} \tag{3.8}$$

Third order terms

$$\begin{aligned} Y_3 = K\frac{b^3}{1-a}(&u_tu_{t-1}u_{t-2} + au_{t-1}u_{t-2}u_{t-3} + a^2u_{t-2}u_{t-3}u_{t-4} + \ldots \\ &+ au_tu_{t-1}u_{t-3} + a^2u_{t-1}u_{t-2}u_{t-4} + \ldots \\ &+ au_tu_{t-2}u_{t-3} + a^2u_{t-1}u_{t-3}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-1}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-2}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-3}u_{t-4} + \ldots \\ &+ \ldots) \end{aligned} \tag{3.9}$$

When an MLB signal is applied, there are coherent discontinuities in the input-output crosscorrelation function. The positions and magnitude of these can be predicted using the shift-and-add property [25, 26] (see Section 2.3.1.2). In Table 3.1, the start positions of the peaks in the crosscorrelation function due to second order terms k (successive terms in (3.8)) and third order terms m (successive terms in (3.9)) are given for three 127-digit MLB signals with different feedback connections.

Start positions of peaks	Feedback stages 4 and 7	Feedback stages 6 and 7	Feedback stages 1, 4, 6 and 7
k_1	97	121	89
$k_2 = 2k_1$	67	115	51
k_3	123	67	21
$k_4 = 4k_1$	7	103	102
k_5	50	78	28
m_1	26	73	59
m_2	115, 46	99, 40	85, 113

Table 3.1. Start positions of peaks in the crosscorrelation function due to second order terms k and third order terms m for three 127-digit MLB signals with different feedback connections.

The crosscorrelation function obtained using a 127-digit MLB signal with feedback from stages 4 and 7 on a process with $T_U = T$ and $T_D = 5T$ is shown in Figure 3.1 (top). The peaks starting at lags 97, 67, 123, etc. due to second order terms and those starting at lags 26, 115, 46, etc. due to third order terms are clearly visible. (The expressions for the crosscorrelation function will be derived in Chapter 5.)

When an inverse-repeat MLB signal was used, the crosscorrelation function was smoother because the effects of even order terms were eliminated. This is shown in Figure 3.1 (middle) using the corresponding inverse-repeat signal of that used in Figure 3.1 (top), and the process has the same dynamics as before. The peaks starting at lags 97, 67, 123, etc. are no longer present. The peak starting at lag 26 is seen to be negative because the third order term is now at lag 153. There is thus a positive discontinuity at lag 153 and a negative discontinuity at lag 26 due to the inverse-repeat property (in which the second half of the crosscorrelation function is the negative of the first half). Positive peaks starting at lags 115 and 46 are still visible. From Figure 3.1 (middle), it can be seen that an inverse-repeat signal is much less suitable for the detection of the departure from linearity using the crosscorrelation function. (It is however more suitable for the estimation of linear dynamics and this will be discussed in Chapter 4.)

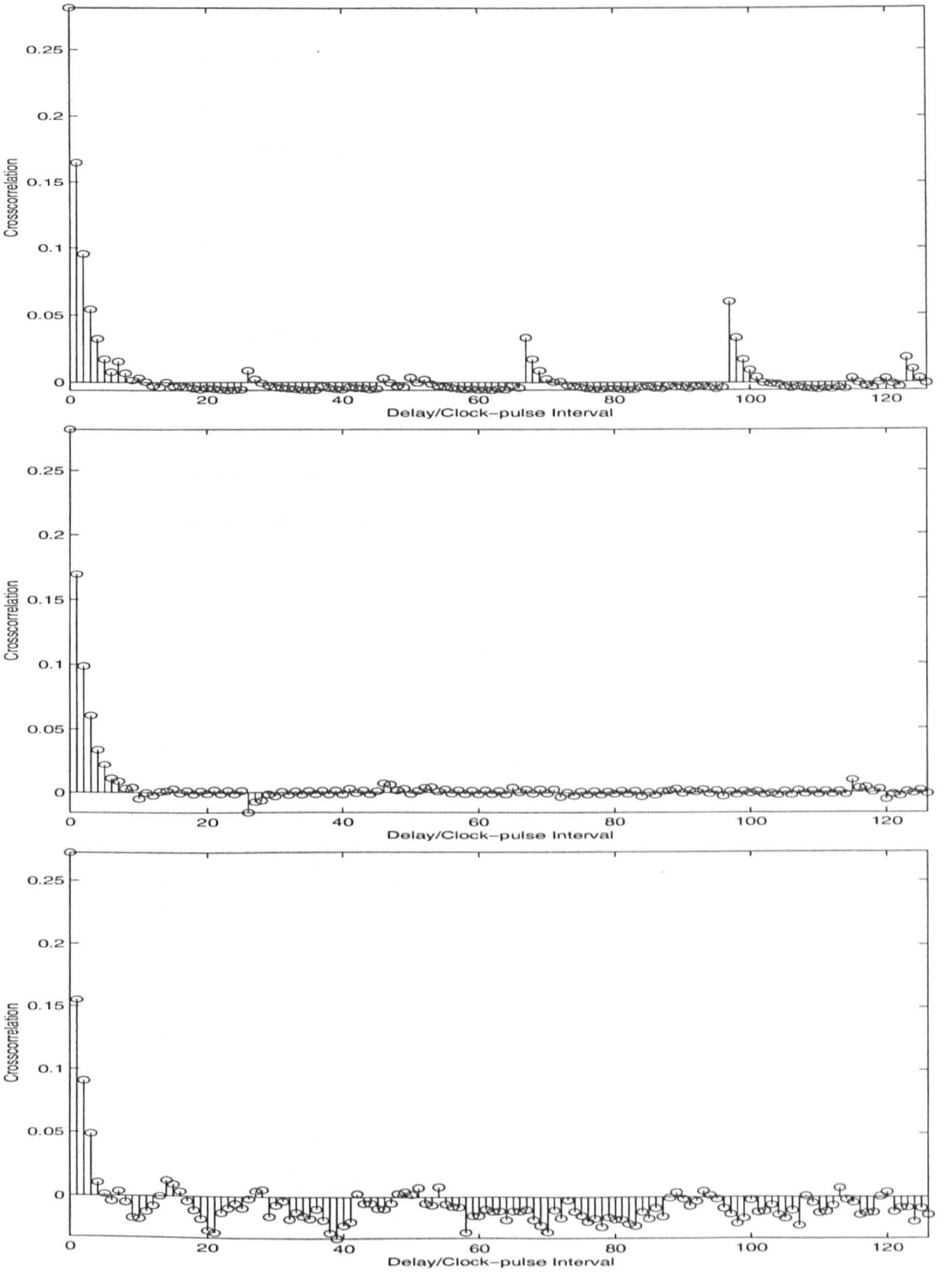

Figure 3.1. Crosscorrelation function for a first order process with dynamics $T_U = T$ and $T_D = 5T$ using three different signals.
Top : 127-digit MLB signal with feedback from stages 4 and 7.
Middle : 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7.
Bottom : 127-digit HAB signal.

Other classes of binary signals do not possess the shift-and-add property. Therefore, there is no coherent pattern in the crosscorrelation functions. This can be seen in Figure 3.1 (bottom) using a 127-digit HAB signal on the same process being considered ($T_U = T$ and $T_D = 5T$). Similar results were obtained using QRB and TPB signals. When their corresponding inverse-repeat signals were used, the crosscorrelation functions were again smoother as predicted.

An important point to note is that the period of the signal used should be significantly larger than the larger time constant of the process in order that coherent patterns in the crosscorrelation function can be easily detected if present. There was no coherent pattern in the frequency response gain or phase when using any of the signals. This is illustrated in Figure 3.2 using the same signal and process as in Figure 3.1 (top).

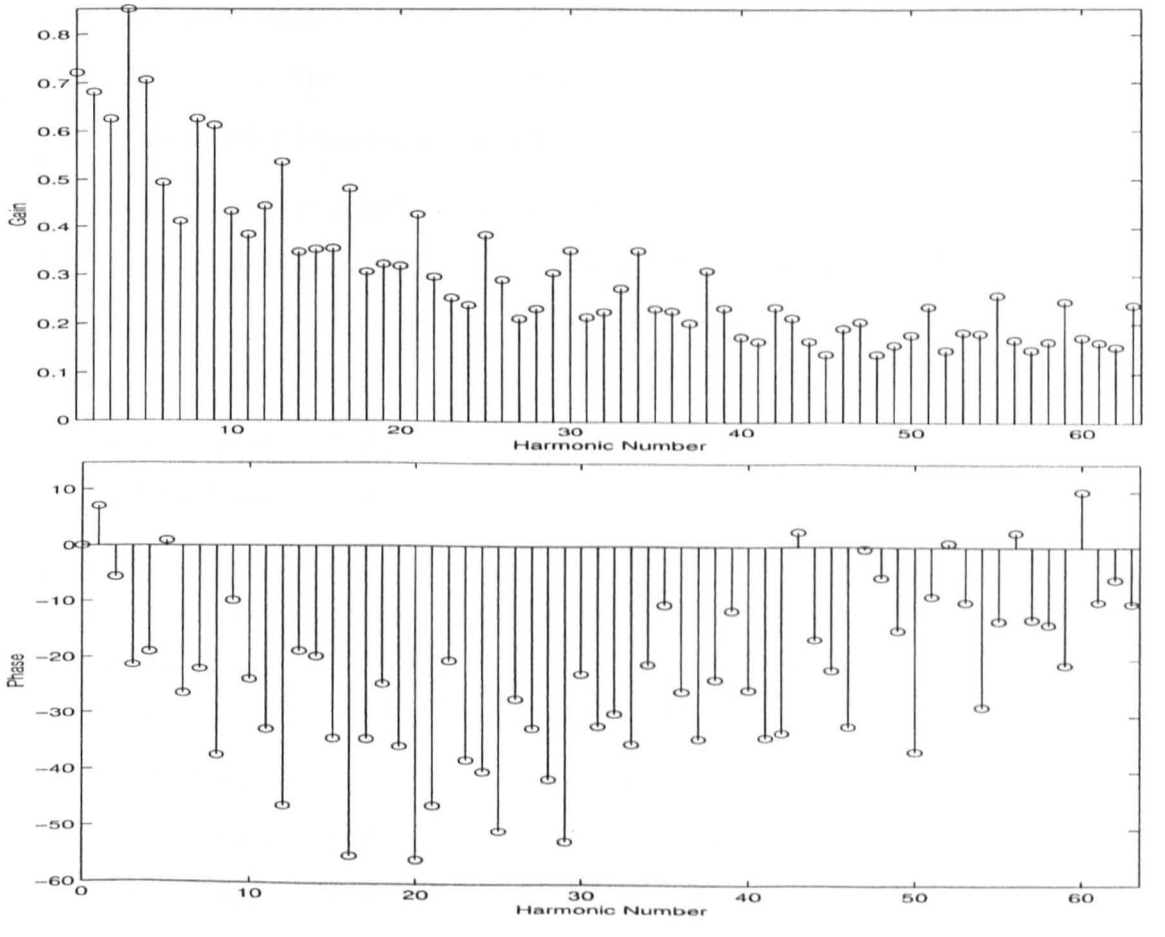

Figure 3.2. Gain response (top) and phase response (bottom) using a 127-digit MLB signal with feedback from stages 4 and 7. The process has dynamics $T_U = T$ and $T_D = 5T$.

3.4.1.2 Detection using Multi-level Maximum Length Signals

MLML sequences [36] with even harmonics suppressed have inverse-repeat properties. These sequences possess the shift-and-subtract property [35]. If a p-level sequence is subtracted, modulo-p, from the same sequence delayed by α digits, the resulting sequence is the original sequence delayed by β digits, where α and β are integers in the range $1 \le \alpha$, $\beta \le p^n - 2$. (This becomes the shift-and-add property for MLB sequences.) Since modulo-p ($p \neq 2$) addition is not equal to modulo-p subtraction, MLML sequences possess the shift-and-add property in addition to the shift-and-subtract property.

For a multi-level signal, analysis using equation (3.3A) is no longer valid. However, there are still coherent peaks in the crosscorrelation function due to the shift-and-add and the shift-and-subtract properties. An example is given in Figures 3.3 to 3.5 for a process with $T_U = T$ and $T_D = 5T$. The input signal is generated from the Galois field (GF(3)) with a primitive polynomial of $1 \oplus_3 2x^4 \oplus_3 x^5$ and a sequence length, N, of $3^5 - 1 = 242$. The more significant third order terms caused by the shifts are given in (3.10). These are arranged such that the resulting shifts lie in the first half of the period.

$$
\begin{aligned}
s_i + s_{i-1} + s_{i-53} &= 0 \\
s_i - s_{i-1} + s_{i-117} &= 0 \\
s_i + s_{i-2} + s_{i-77} &= 0 \\
s_i - s_{i-2} + s_{i-49} &= 0 \\
-s_i - s_{i-3} + s_{i-38} &= 0 \\
s_i - s_{i-3} + s_{i-109} &= 0 \\
-s_i - s_{i-4} + s_{i-57} &= 0 \\
s_i - s_{i-4} + s_{i-5} &= 0 \\
-s_i - s_{i-5} + s_{i-4} &= 0 \\
s_i - s_{i-5} + s_{i-57} &= 0
\end{aligned}
\qquad
\begin{aligned}
s_i + s_{i-1} + s_{i-2} + s_{i-113} &= 0 \\
-s_i - s_{i-1} + s_{i-2} + s_{i-35} &= 0 \\
-s_i + s_{i-1} - s_{i-2} + s_{i-106} &= 0 \\
-s_i + s_{i-1} + s_{i-2} + s_{i-22} &= 0 \\
-s_i - s_{i-1} - s_{i-3} + s_{i-18} &= 0 \\
-s_i - s_{i-1} + s_{i-3} + s_{i-15} &= 0 \\
s_i - s_{i-1} + s_{i-3} + s_{i-12} &= 0 \\
s_i - s_{i-1} - s_{i-3} + s_{i-88} &= 0 \\
-s_i - s_{i-2} - s_{i-3} + s_{i-31} &= 0 \\
s_i + s_{i-2} - s_{i-3} + s_{i-75} &= 0 \\
-s_i + s_{i-2} - s_{i-3} + s_{i-47} &= 0 \\
-s_i + s_{i-2} + s_{i-3} + s_{i-96} &= 0
\end{aligned}
\qquad (3.10)
$$

It should also be noted that terms that are apparently second order are indeed third order since

$$a_1 s_{i-J} + a_2 s_{i-K} + s_{i-I} = 0 \qquad \text{mod-3} \qquad (3.11)$$

can be written in three distinct forms [29]

$$\begin{aligned} -a_1 s_{i-J} - a_2 s_{i-K} + s_{i-I} + s_{i-I} &= 0 \\ a_1 s_{i-J} - a_2 s_{i-K} - a_2 s_{i-K} + s_{i-I} &= 0 \qquad \text{mod-3} \qquad (3.12) \\ -a_1 s_{i-J} - a_1 s_{i-J} + a_2 s_{i-K} + s_{i-I} &= 0 \end{aligned}$$

where a_1 and a_2 are elements of GF(3), and i, J, K and I are integers.

Some of the peaks predicted by (3.10) can be clearly observed in Figure 3.5 such as the negative peaks starting at lags 53 and 77, and the positive peaks starting at lags 117 and 38. The sign and magnitude of the resulting peaks depend on the signs of the terms in (3.10) and the magnitude of the impulse response at times corresponding to the lags in (3.10). Unfortunately, detailed theoretical analysis could not be done due to equation (3.3A) not being valid.

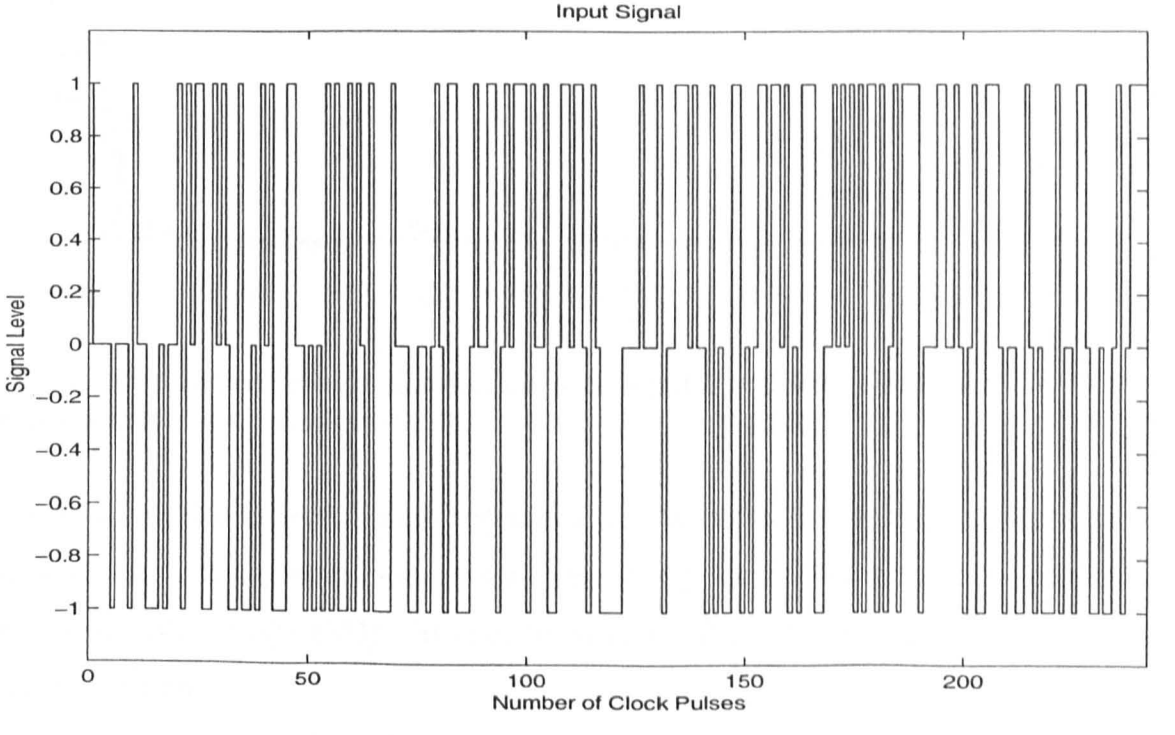

Figure 3.3. Input signal with GF(3) and $n = 5$.

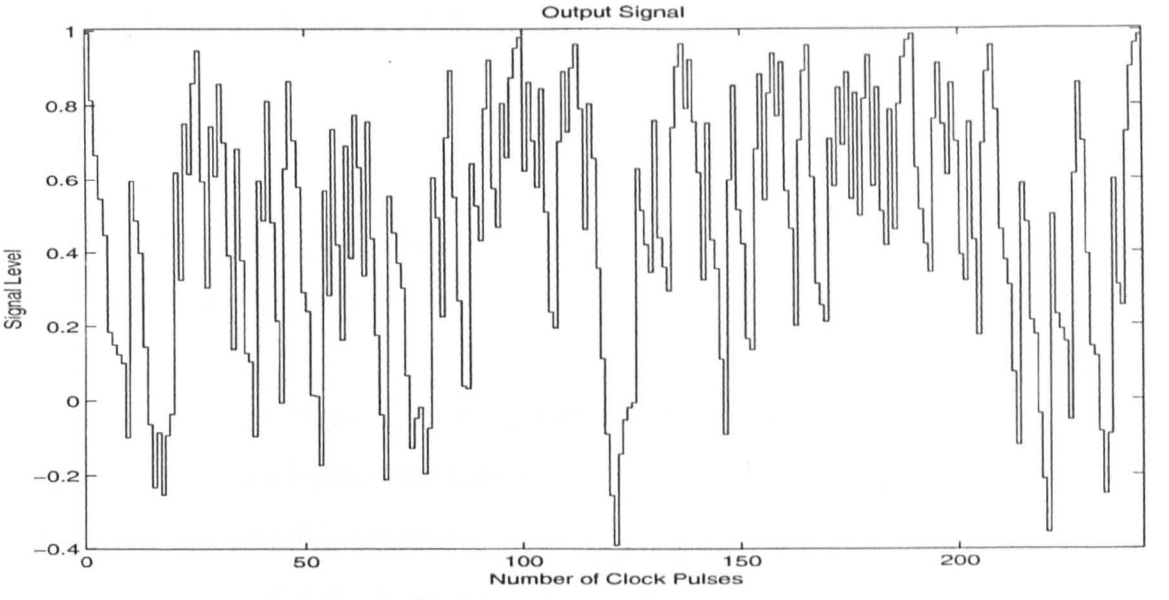

Figure 3.4. Output using an input signal with GF(3) and $n = 5$ for a process with $T_U = T$ and $T_D = 5T$.

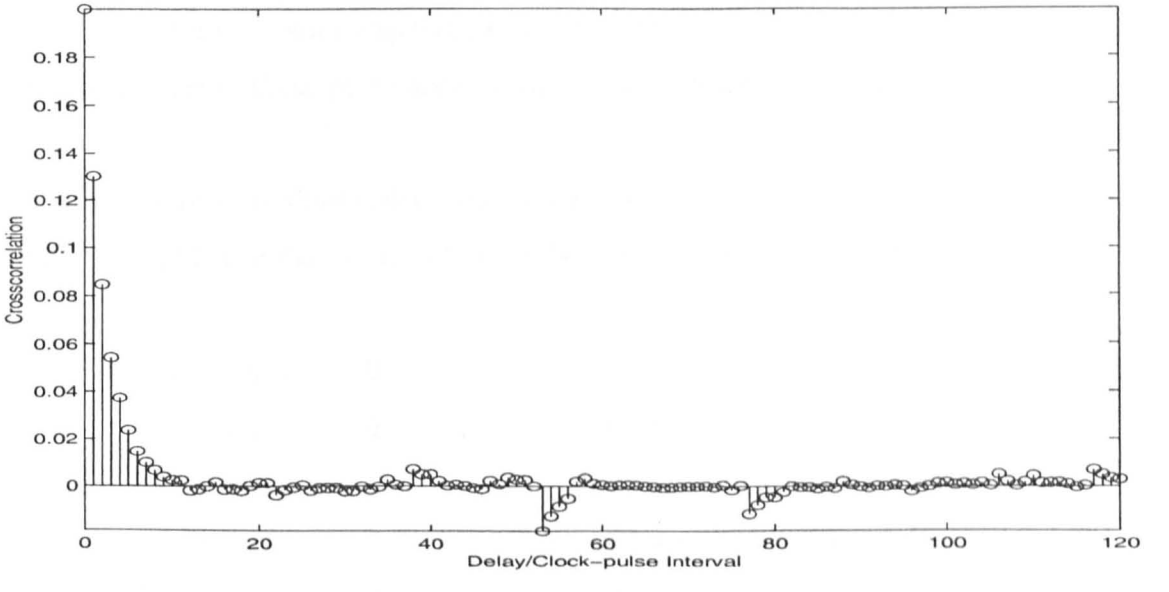

Figure 3.5. Crosscorrelation function using an input signal with GF(3) and $n = 5$ for a process with $T_U = T$ and $T_D = 5T$.

For multi-level maximum length sequences with harmonic multiples of two and three suppressed, both the inverse-repeat requirement $u(i) + u(i+N/2) = 0$ and the constraint $u(i) + u(i+N/3) + u(i+2N/3) = 0$ must be fulfilled. The autocorrelation function $R_{uu}(i)$ has the property

$$R_{uu}(i) + R_{uu}(i+N/3) + R_{uu}(i+2N/3) = 0 \qquad \text{for all } i \qquad (3.13)$$

This leads to the fact that in a normalised autocorrelation,

$$
\begin{aligned}
&R_{uu}(0) = 1 \\
&R_{uu}(N/2) = -1 \\
&R_{uu}(N/6) = R_{uu}(5N/6) = 0.5 \\
&R_{uu}(N/3) = R_{uu}(2N/3) = -0.5
\end{aligned} \tag{3.14}
$$

These peaks can be observed in the crosscorrelation functions such as that shown in Figure 3.8 of a signal generated from GF(7) with $n = 3$ and the primitive polynomial $1 \oplus_7 3x^2 \oplus_7 2x^3$. For this sequence, $N = 7^3 - 1 = 342$. The process has time constants $T_U = T$ and $T_D = 5T$. (The input and output signals are shown in Figures 3.6 and 3.7 respectively.) A large positive peak starts at lag zero and smaller positive peaks start at lags $(N / 6) = 57$ and $(5N / 6) = 285$. Similarly, a large negative peak starts at lag $(N / 2) = 171$ and smaller negative peaks start at lags $(N / 3) = 114$ and $(2N / 3) = 228$. The amplitudes of these peaks are distorted by the departure from linearity.

The more significant third order terms caused by the shifts are given in (3.15). These are again arranged such that the resulting shifts lie in the first half of the period.

$$
\begin{aligned}
&-s_i - s_{i-1} + s_{i-63} = 0 \\
&s_i - s_{i-1} + s_{i-22} = 0 \\
&-s_i - s_{i-2} + s_{i-8} = 0 \\
&s_i - s_{i-2} + s_{i-85} = 0 \\
&s_i + s_{i-3} + s_{i-47} = 0 \\
&s_i - s_{i-3} + s_{i-122} = 0 \\
&s_i + s_{i-4} + s_{i-80} = 0 \\
&s_i - s_{i-4} + s_{i-93} = 0 \\
&-s_i - s_{i-5} + s_{i-139} = 0 \\
&-s_i + s_{i-5} + s_{i-101} = 0
\end{aligned}
\qquad
\begin{aligned}
&-s_i - s_{i-1} - s_{i-2} + s_{i-100} = 0 \\
&-s_i - s_{i-1} + s_{i-2} + s_{i-35} = 0 \\
&-s_i + s_{i-1} - s_{i-2} + s_{i-155} = 0 \\
&-s_i + s_{i-1} + s_{i-2} + s_{i-11} = 0 \\
&s_i + s_{i-1} + s_{i-3} + s_{i-20} = 0 \\
&s_i + s_{i-1} - s_{i-3} + s_{i-88} = 0 \\
&-s_i + s_{i-1} - s_{i-3} + s_{i-146} = 0 \\
&s_i - s_{i-1} - s_{i-3} + s_{i-49} = 0 \\
&-s_i - s_{i-2} - s_{i-3} + s_{i-142} = 0 \\
&s_i + s_{i-2} - s_{i-3} + s_{i-104} = 0 \\
&-s_i + s_{i-2} - s_{i-3} + s_{i-17} = 0 \\
&s_i - s_{i-2} - s_{i-3} + s_{i-81} = 0
\end{aligned} \tag{3.15}
$$

From Figure 3.8, positive peaks can be observed starting at lags 63 and 8, while negative peaks can be observed starting at lags 100 and 11. It is interesting to note that in the case of MLML sequences with harmonic multiples of two and three suppressed, the linear terms are large and the peaks caused by higher order terms become less noticeable.

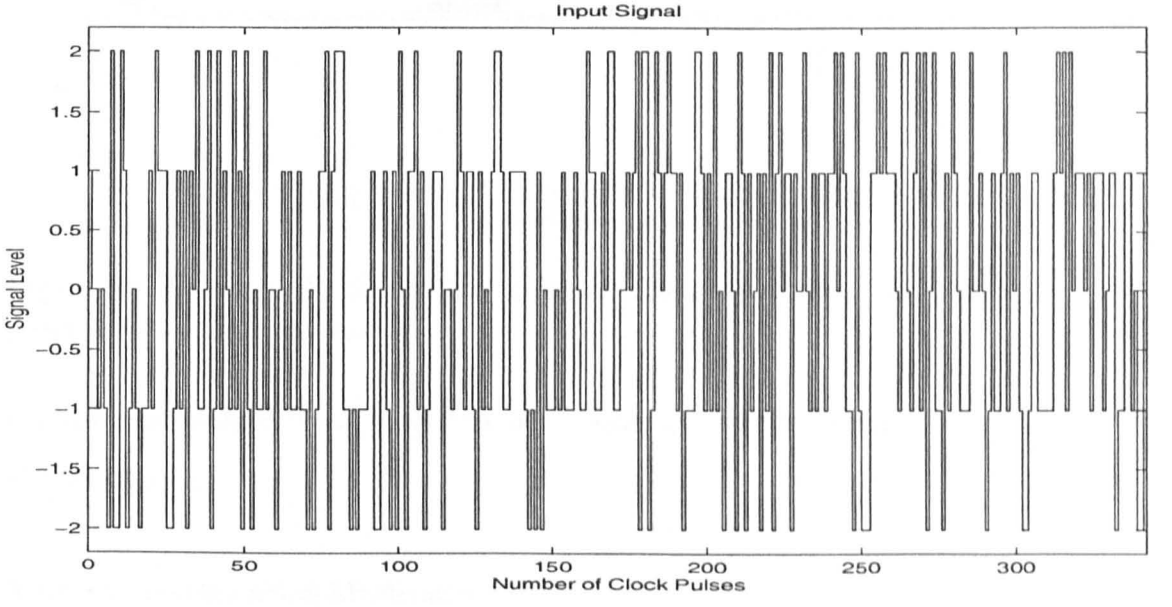

Figure 3.6. Input signal with GF(7) and $n = 3$.

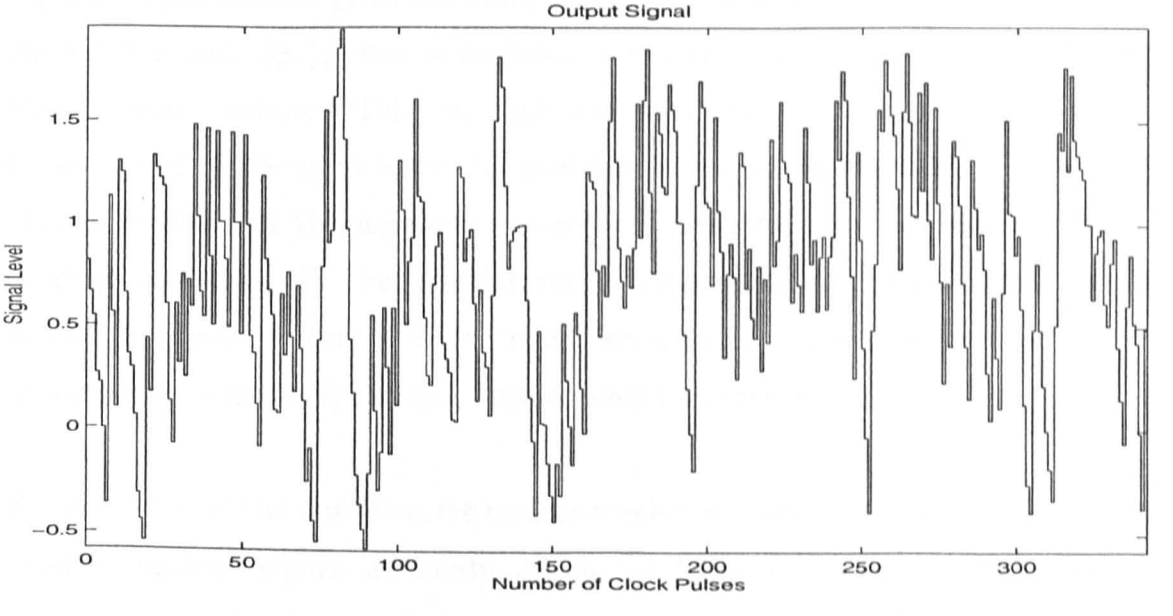

Figure 3.7. Output using an input signal with GF(7) and $n = 3$ for a process with $T_U = T$ and $T_D = 5T$.

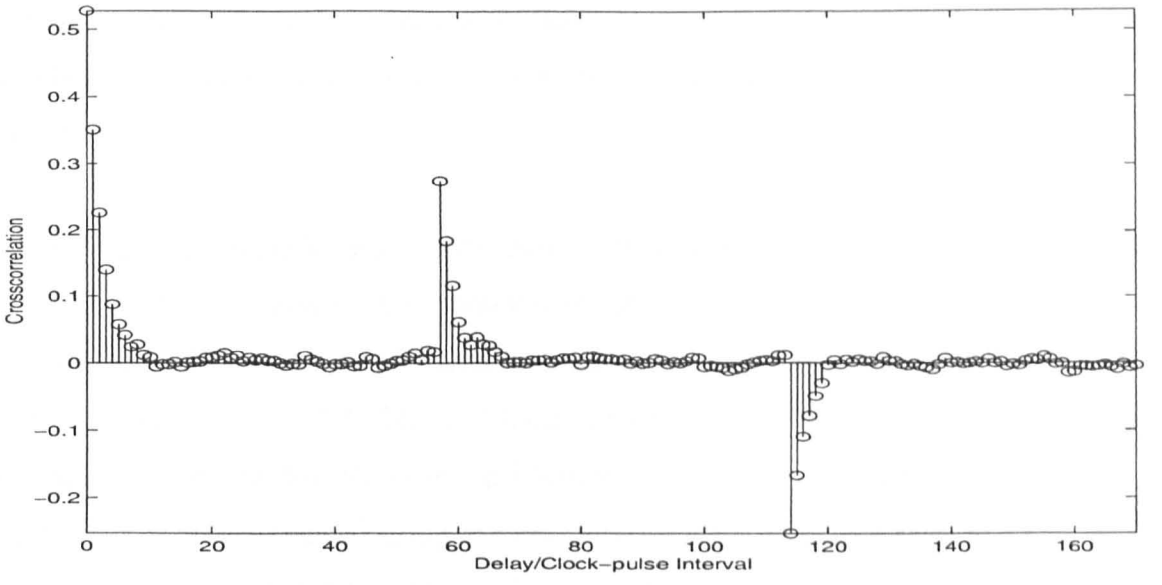

Figure 3.8. Crosscorrelation function using an input signal with GF(7) and $n = 3$ for a process with $T_U = T$ and $T_D = 5T$.

No coherent pattern was present in the frequency domain when using either of the MLML signals.

3.4.1.3 Detection using Multisines

Multisine signals can be generated using the function *msinzoh* which is a modification of the function *msinclip* [5] that is available in the MATLAB Frequency Domain System Identification Toolbox. (This function was modified by Andrew Tucker of the University of Warwick to allow the generated signal to be loaded directly into the MATLAB workspace.) The algorithm minimises the crest factor of the multisine using a clipping algorithm [44]. The time domain waveform is clipped and then transformed into the frequency domain. The amplitudes are restored to the desired values, and the clipping level is gradually reduced, thus reducing the signal amplitude.

To generate a random multisine, the random number generator in MATLAB can be used to generate random phases uniformly distributed between 0 and 2π. The complex amplitudes at the specified nonzero frequencies are entered into *msinzoh*. To generate a Schroeder multisine [43] instead, the amplitudes at the specified nonzero frequencies are

entered as real values. In order not to have the crest factor minimised, the iteration number should be set equal to one. The signal generated then has an almost Gaussian distribution.

Three harmonic specifications were used in this application. The first was the odd multisine with excitation lines at harmonic numbers $f = 1, 3, 5, 7, 9, 11$, etc. The second was the odd-odd multisine with excitation lines at $f = 1, 5, 9, 13$, etc. while the third had excitation lines at $f = 1, 3, 9, 11, 17, 19$, etc. There was no coherent pattern in the output frequency spectrum for all three specifications, either using random or Schroeder multisines. Results obtained using the specification with excitation lines at $f = 1, 3, 9, 11, 17, 19$, etc. are shown in Figures 3.9 to 3.14. Figures 3.9 and 3.10 show the random multisine and the process output respectively. The process has time constants $T_U = T$ and $T_D = 5T$. Figures 3.11 and 3.12 show the discrete Fourier transform of the input and output respectively. The Schroeder multisine is shown in Figure 3.13 and the discrete Fourier transform of the output is given in Figure 3.14 for the same process. Similar results were obtained for the odd and odd-odd multisines (in that no coherent pattern was observed in the output frequency spectrum).

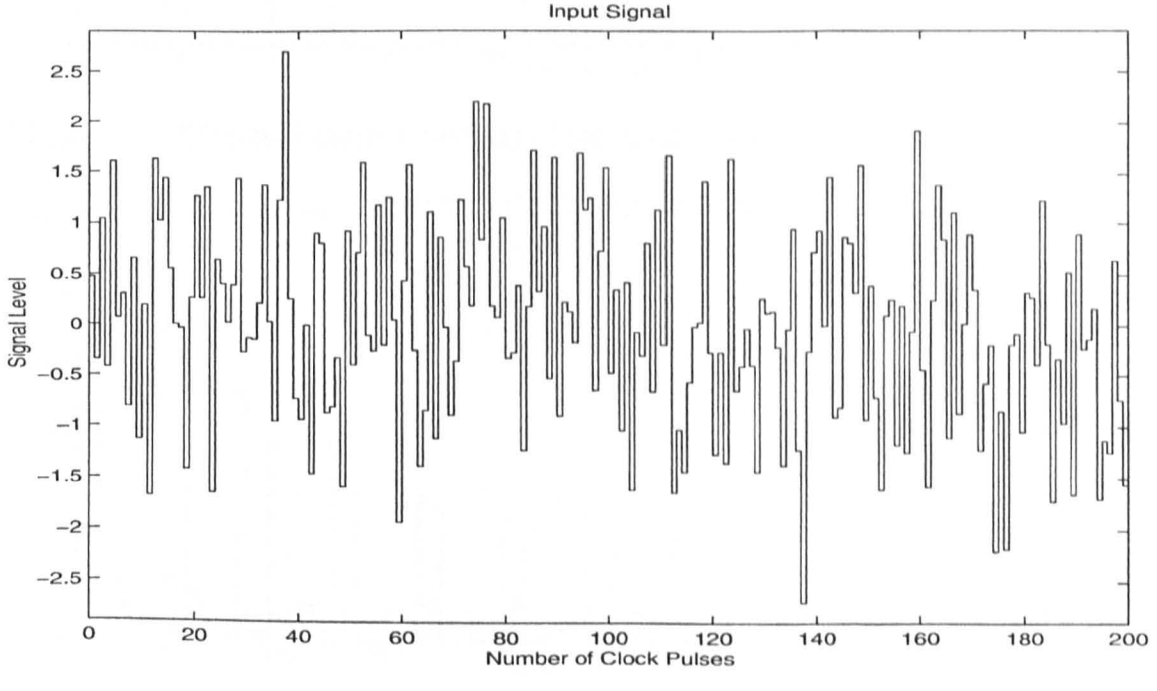

Figure 3.9. Random multisine with excitation lines at $f = 1, 3, 9, 11, 17, 19$, etc.

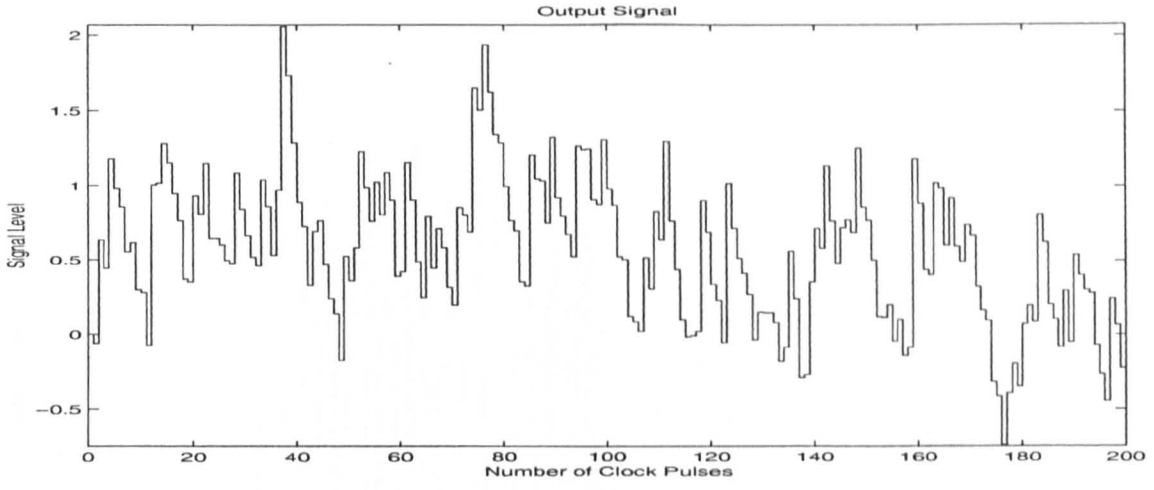

Figure 3.10. Output using a random multisine with excitation lines at $f = 1, 3, 9, 11, 17, 19$, etc. The process has time constants $T_U = T$ and $T_D = 5T$.

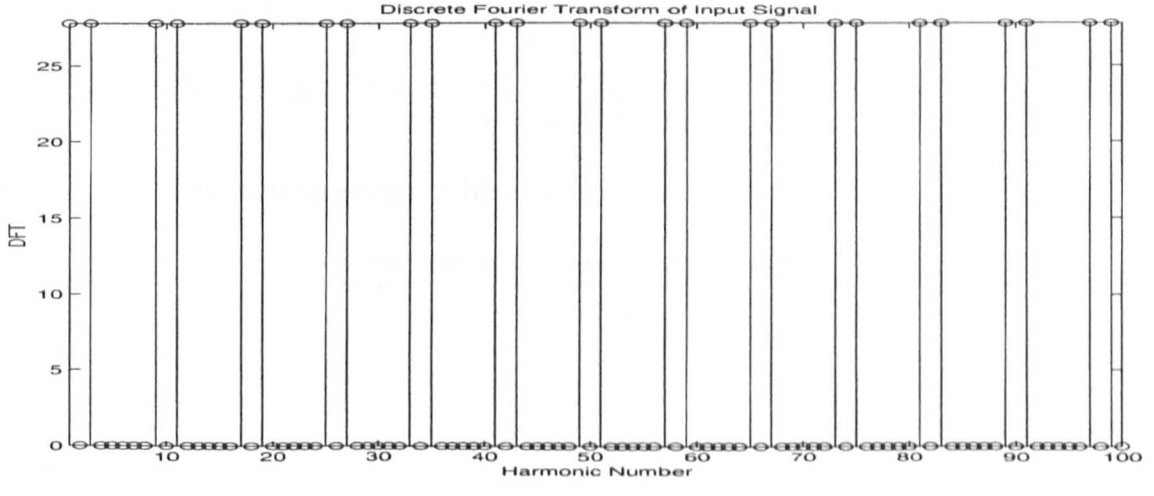

Figure 3.11. Discrete Fourier transform of the input signal.

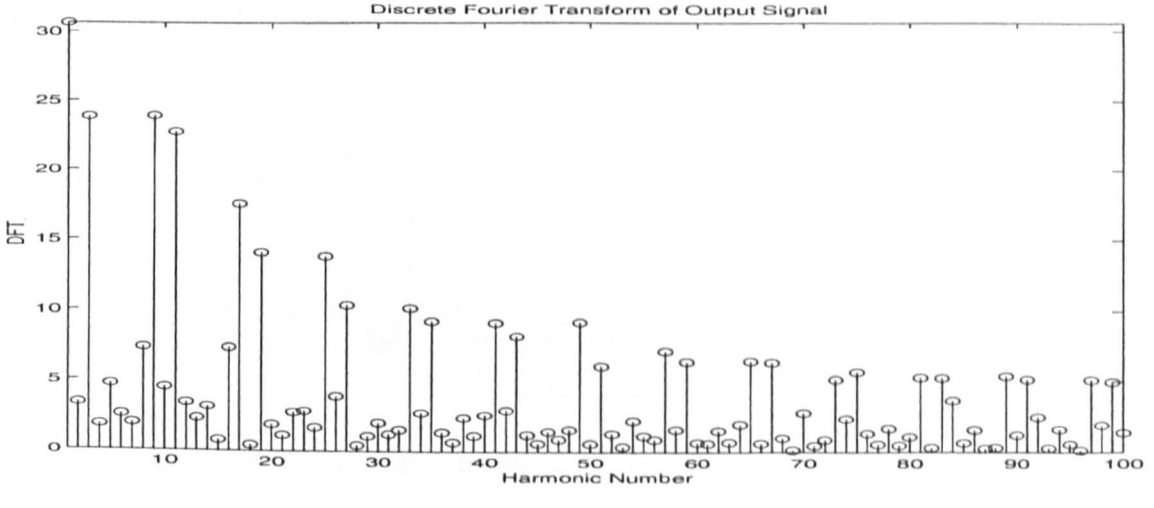

Figure 3.12. Discrete Fourier transform of the output signal.

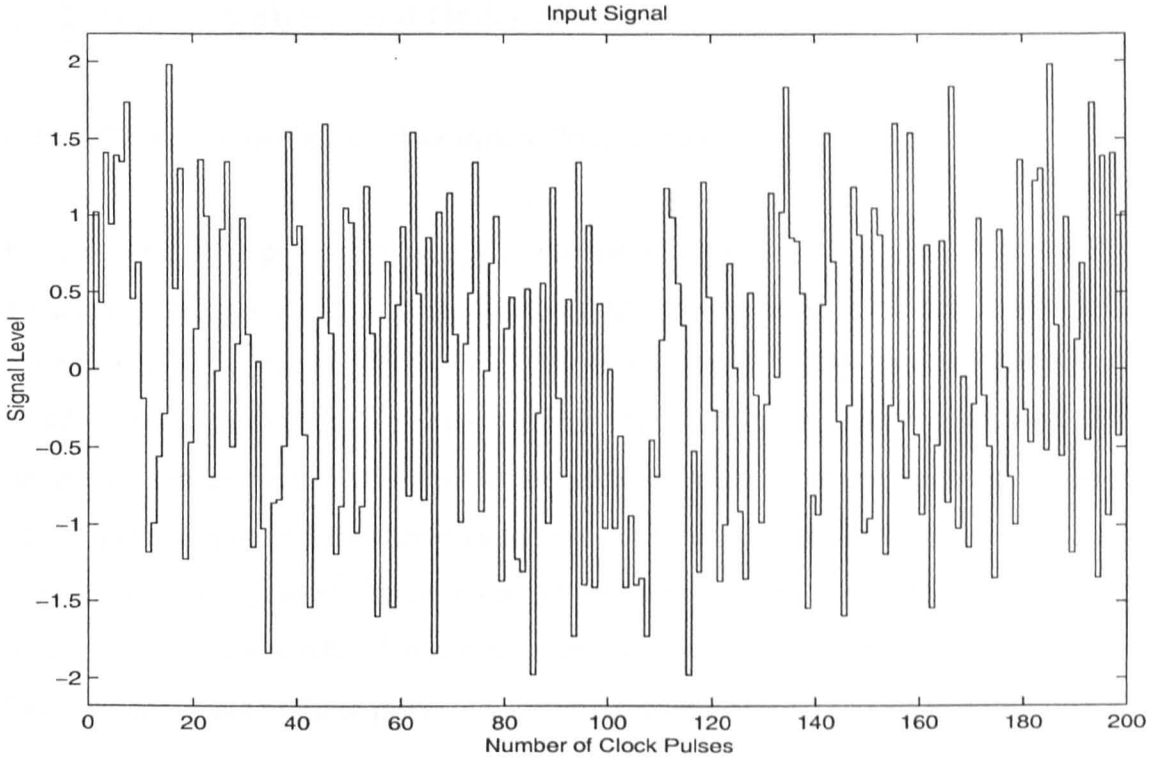

Figure 3.13. Schroeder multisine with excitation lines at $f = 1, 3, 9, 11, 17, 19$, etc.

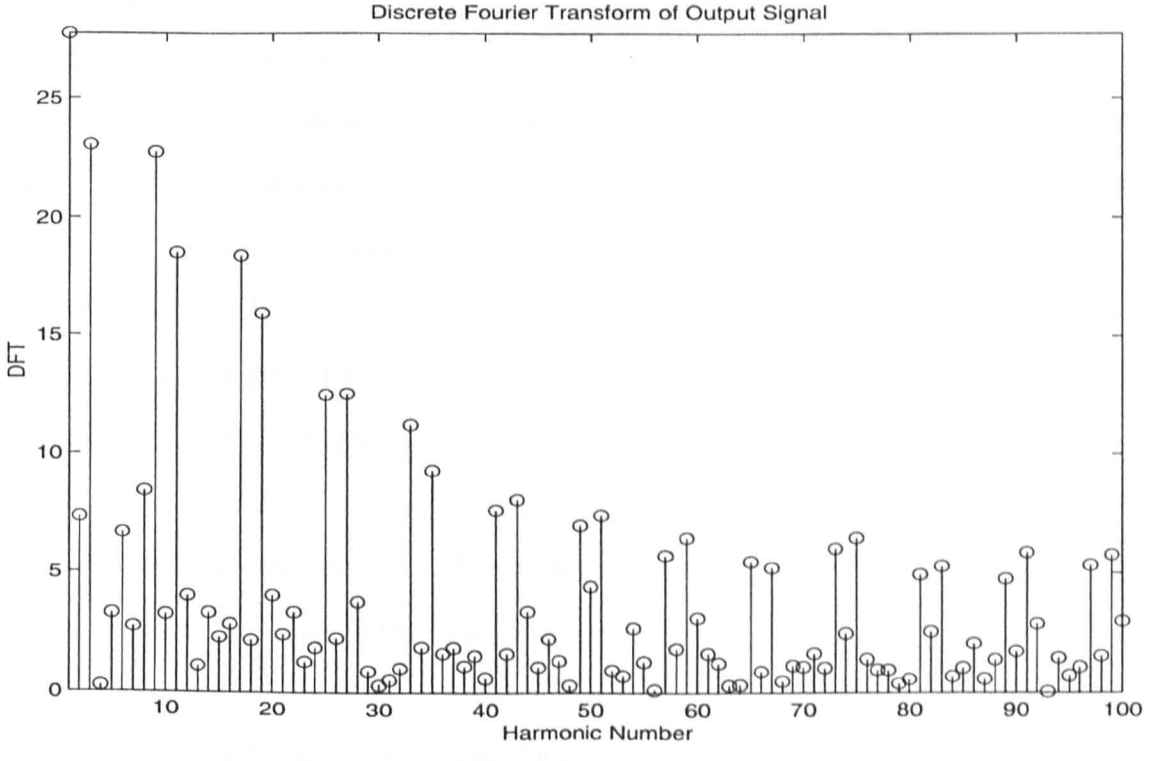

Figure 3.14. Discrete Fourier transform of the output signal using the Schroeder multisine.

3.4.2 Process with Second Order Dynamics

3.4.2.1 Detection using Pseudo-random Binary Signals

For a second order process, the slope of the output y no longer responds immediately to a change in input u. In order to predict the output y reasonably accurately, the sampling interval was set to twenty times the clock-pulse interval. Due to the fact that equation (3.3A) is no longer valid, the results obtained are much more difficult to analyse. When an MLB signal is used, there are peaks in the crosscorrelation function in addition to that of the linear response. It is necessary to look for peaks starting in the same positions as for the first order process (those given by the shift-and-add property). However, these peaks are not necessarily of the same sign as that obtained for the first order case. Furthermore, some of these peaks may be undetectable due to their small amplitude.

Examples are given in Figures 3.15 to 3.17 using a 127-digit MLB signal with feedback from stages 1, 4, 6 and 7 (the second order terms k and the third order terms m for this signal are given in Table 3.1). Figure 3.15 shows overdamped processes, with damping factor $\zeta = 2$ in both directions. In Figure 3.15 (top), the undamped natural frequency is $\omega_{nU} = 5/T$ in the upward direction (which corresponds to $T_U = 0.05T$, $0.75T$) and $\omega_{nD} = 1.25/T$ in the downward direction (which corresponds to $T_D = 0.21T$, $2.99T$). In Figure 3.15 (middle), $\omega_{nU} = 3/T$ ($T_U = 0.09T$, $1.24T$) and $\omega_{nD} = 0.75/T$ ($T_D = 0.36T$, $4.98T$) while in Figure 3.15 (bottom), $\omega_{nU} = 1/T$ ($T_U = 0.27T$, $3.73T$) and $\omega_{nD} = 0.25/T$ ($T_D = 1.07T$, $14.93T$). Figure 3.16 shows a critically damped process with $\zeta = 1$ in both directions, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_U = 4T$, $4T$). Figure 3.17 shows an underdamped process with $\zeta = 0.25$ in both directions, $\omega_{nU} = 1/T$ and $\omega_{nD} = 0.25/T$; there are no real time constants in this case since $\zeta < 1$.

From these figures it is seen that there are still coherent discontinuities in the crosscorrelation functions, but not all of those observed in the first order case are necessarily present, or necessarily of the same sign. For example, the first second order

term at lag 89 causes a positive discontinuity in Figure 3.15 (top), is undetectable in Figure 3.15 (middle) and causes a negative discontinuity in Figure 3.15 (bottom).

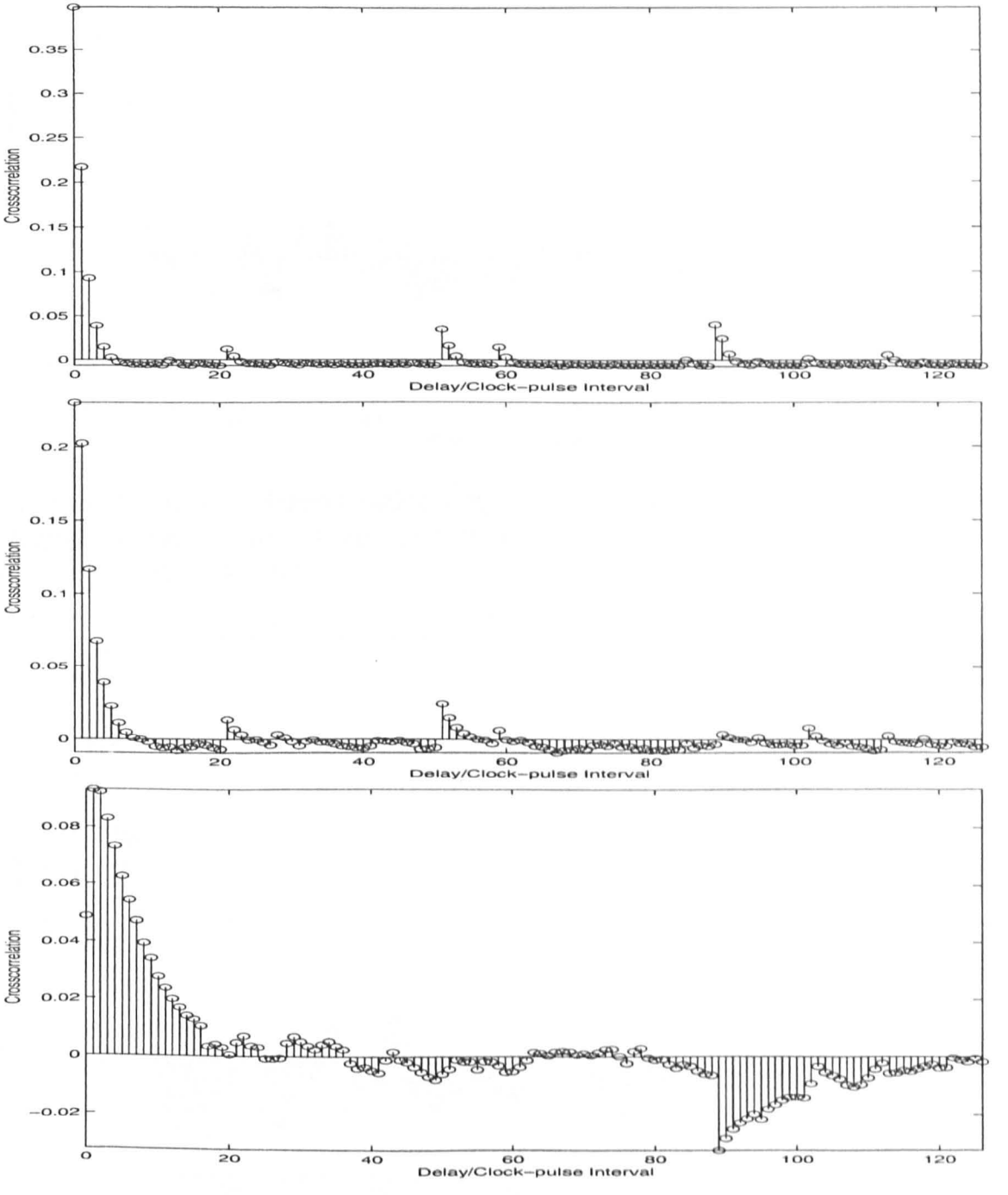

Figure 3.15. Crosscorrelation function for a second order process with $\zeta_U = \zeta_D = 2$ using a 127-digit MLB signal with feedback from stages 1, 4, 6 and 7.
Top : $\omega_{nU} = 5/T$ ($T_U = 0.05T$, $0.75T$) and $\omega_{nD} = 1.25/T$ ($T_D = 0.21T$, $2.99T$).
Middle : $\omega_{nU} = 3/T$ ($T_U = 0.09T$, $1.24T$) and $\omega_{nD} = 0.75/T$ ($T_D = 0.36T$, $4.98T$).
Bottom : $\omega_{nU} = 1/T$ ($T_U = 0.27T$, $3.73T$) and $\omega_{nD} = 0.25/T$ ($T_D = 1.07T$, $14.93T$).

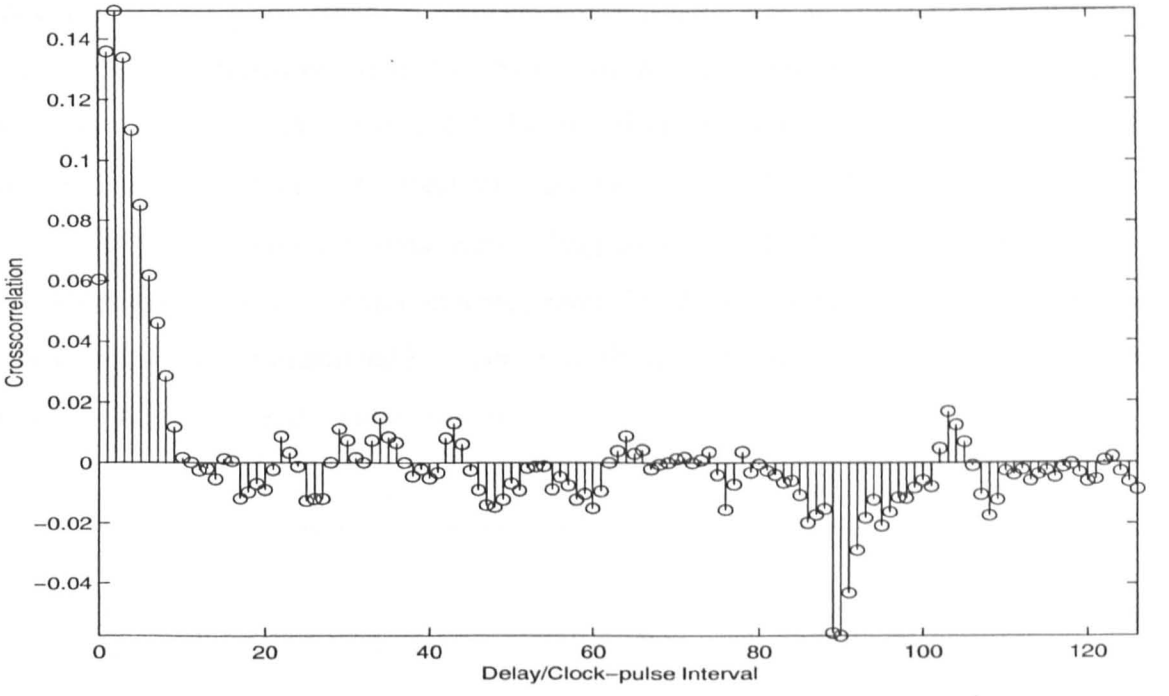

Figure 3.16. Crosscorrelation function using a 127-digit MLB signal with feedback from stages 1, 4, 6 and 7. Second order process with $\zeta_U = \zeta_D = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_D = 4T$, $4T$).

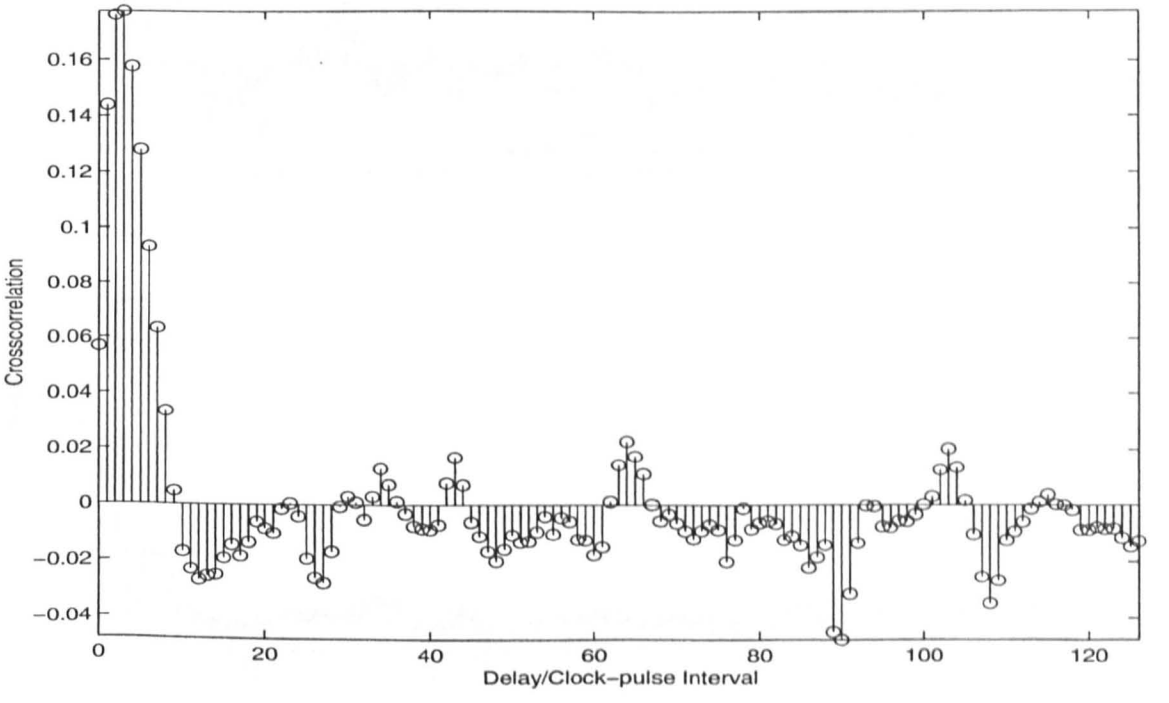

Figure 3.17. Crosscorrelation function using a 127-digit MLB signal with feedback from stages 1, 4, 6 and 7. Second order process with $\zeta_U = \zeta_D = 0.5$, $\omega_{nU} = 1/T$ and $\omega_{nD} = 0.25/T$.

When other classes of binary signals were used, there was no coherent pattern in the crosscorrelation function due to the absence of the shift-and-add property. This can be seen in Figure 3.18 (top) using a 127-digit QRB signal and the process is critically damped with $\zeta = 1$ in both directions, and $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_U = 4T$, $4T$). Similar results were obtained using HAB and TPB signals. The crosscorrelation functions using inverse-repeat signals were again smoother as predicted from theory. This is illustrated in Figure 3.18 (bottom) using the corresponding inverse-repeat QRB signal on the same process.

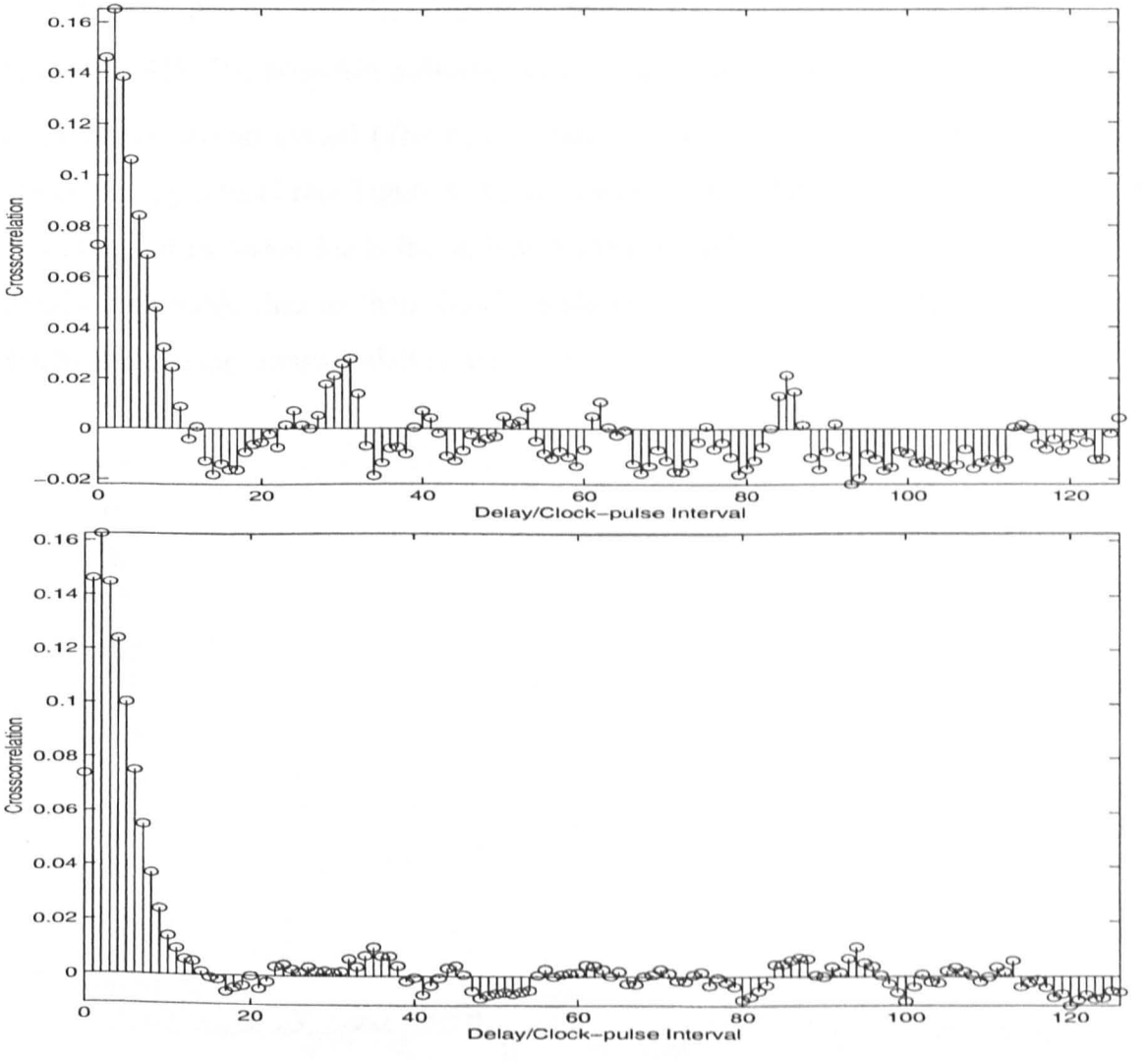

Figure 3.18. Crosscorrelation function for a second order process with $\zeta_U = \zeta_D = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_D = 4T$, $4T$) using two different signals.
Top : 127-digit QRB signal.
Bottom : 254-digit inverse-repeat QRB signal.

3.4.2.2 Detection using Multi-level Maximum Length Signals

As in the first order case, the shift-and-add and the shift-and-subtract properties caused some peaks in the crosscorrelation function. However, many of them were not readily detectable due to their small amplitude (compared to the peaks caused by an MLB signal on the same process). The directions of these peaks could not be predicted as when using an MLB signal on a second order process.

Figure 3.19 shows the crosscorrelation function using an input signal with GF(3) and $n = 5$ for a process with $\zeta_U = \zeta_D = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_D = 4T$, $4T$). The primitive polynomial of the input signal is $1 \oplus_3 2x^4 \oplus_3 x^5$ and it has even harmonics suppressed. (The signal is plotted in Figure 3.3 and the third order shifts are given in (3.10).) From Figure 3.19, the discontinuity at lag 53 is clearly visible. The other third order terms due to the shift-and-add and shift-and-subtract properties are not readily detectable due to their small amplitude (compared to the amplitude of the fluctuations in the crosscorrelation function).

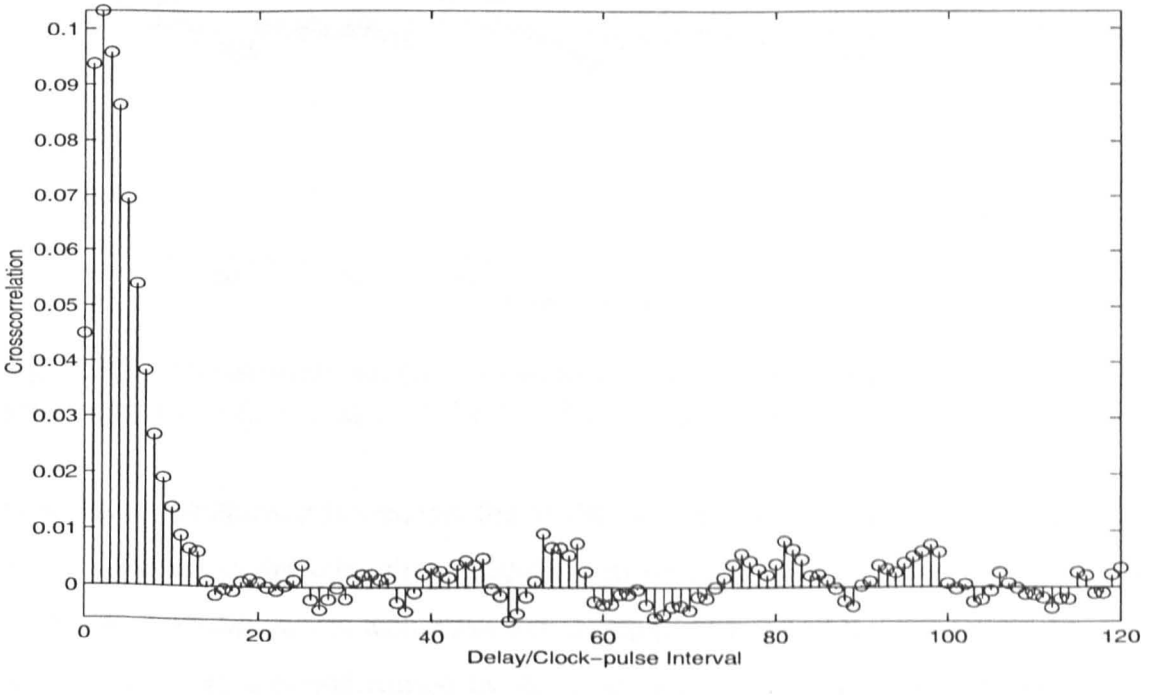

Figure 3.19. Crosscorrelation function using an input signal with GF(3) and $n = 5$ for a process with $\zeta_U = \zeta_D = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_D = 4T$, $4T$).

Figure 3.20 shows the crosscorrelation function using an input signal with GF(7) and $n = 3$ for the same process. The primitive polynomial of the input signal is $1 \oplus_7 3x^2 \oplus_7 2x^3$ and it has harmonic multiples of two and three suppressed. (The signal is plotted in Figure 3.6 and the third order shifts are given in (3.15).) From Figure 3.20, the linear terms due to the autocorrelation property given by equation (3.13) are again present. The only other discontinuity which is readily detectable starts at lag 63. The amplitude of the discontinuities caused by third order terms are small compared to the linear response.

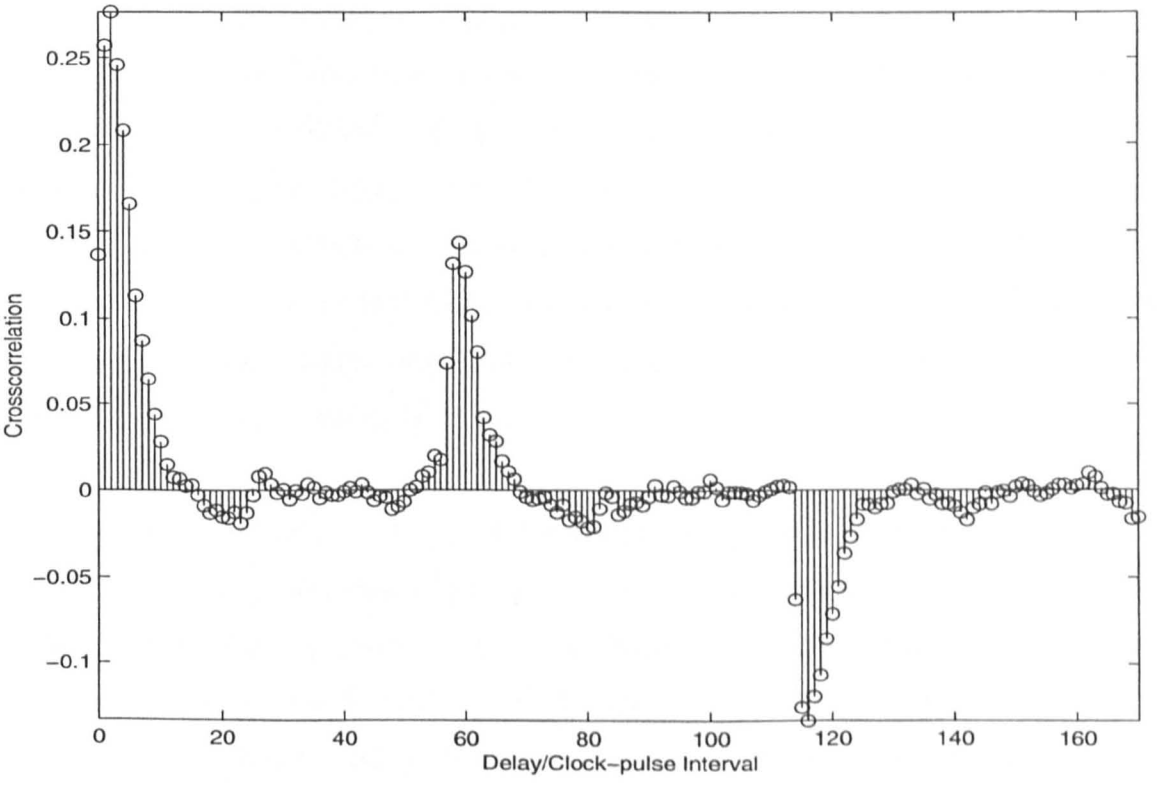

Figure 3.20. Crosscorrelation function using an input signal with GF(7) and $n = 3$ for a process with $\zeta_U = \zeta_D = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_D = 4T$, $4T$).

From these examples, it is seen that the MLML signal is much less suitable compared to the MLB signal in detecting the departure from linearity for a second order process. In particular, a MLML signal with harmonic multiples of two and three suppressed has six discontinuities in a period caused by the linear terms. This makes the discontinuities caused by higher order terms much more difficult to detect.

3.4.2.3 Detection using Multisines

As in the case of a first order process, there was no coherent pattern in the input-output crosscorrelation function and the output frequency spectrum when either random or Schroeder multisines were used.

3.5 Conclusions

Pseudo-random signals and multisines were applied to processes with direction-dependent dynamics and with the dynamics linear in both directions of the output. To detect the presence of this type of departure from linearity, an MLB signal should be applied and the identification carried out in the time domain as there is no coherent pattern in the frequency domain. For a first order process, coherent patterns in the input-output crosscorrelation function can be predicted theoretically as a result of the shift-and-add property of the MLB signal. The period of the perturbation signal should be significantly larger than the largest time constant of the process so that these coherent patterns can be easily detected if present.

For a second order process, analytical expressions could not be obtained for the process output as the sign of the slope of the output is not necessarily the same as the sign of the input (assuming that the input levels are symmetrical about zero). It is possible to detect the departure from linearity using an MLB signal as there are still coherent patterns in the crosscorrelation function as in the first order case. However, these discontinuities are not necessarily of the same sign as for a process with first order dynamics and some of them may be undetectable due to their small amplitudes. The use of an inverse-repeat signal eliminates the effects of even order nonlinearities making the crosscorrelation function smoother. Thus, it is much less suitable for the purpose of detecting the departure from linearity using the crosscorrelation function. Other classes of pseudo-random binary signals do not possess the shift-and-add property and hence no predictable pattern can be found in the crosscorrelation function.

When a MLML signal is used, the output of the process no longer increases or decreases depending on the sign of the input. For a first order process, there are some discontinuities in the crosscorrelation function due to the shift-and-add and the shift-and-subtract properties. For a second order process, most of these discontinuities are not readily detectable due to their small amplitude (compared to when using an MLB signal). This makes the MLML signal much less suitable for detecting the direction-dependent behaviour especially if the signal has harmonic multiples of two and three suppressed (as the occurrence of more linear terms in a period will mask the effects of higher order terms).

There is no coherent pattern in the frequency response gain or phase when using any of the pseudo-random and multisine signals. This makes the multisine signals totally unsuitable for the detection of the direction-dependent characteristics (since there is no coherent pattern in the input-output crosscorrelation function either).

Chapter 4

Estimation of Linear Dynamics for Processes with Direction-dependent Dynamics

4.1 Introduction

Pseudo-random signals can be applied to estimate the parameters of a linear model of processes with direction-dependent responses and with the dynamics linear in both directions. For the first order case, the combined time constant can be obtained through simple calculations, and this is found to be closer to the smaller time constant of the process. If this linear model is the only one obtained for such a process, this could present a considerable control problem, because a controller designed for the estimated dynamics might function poorly in the slower direction, leading to possible instability in extreme cases. This demonstrates the importance of recognising the direction-dependent behaviour of the process (as was described in Chapter 3 using an MLB perturbation signal) because then, the controller parameters could be designed to cope with such a behaviour in the process response. It is also possible to obtain a linear model for both first and second order processes using parameter estimation algorithms available in the MATLAB System Identification Toolbox [45] and the MATLAB Frequency Domain System Identification Toolbox [5], and these are based on parametric identification methods. A nonparametric identification method using correlation analysis [46, 47] can be used for the same purpose. An overview of the model structures used and a comparison of the results obtained by different algorithms are given. For both first and second order processes, the overall gain is also estimated for a process in which the gain is direction-dependent.

4.2 Parameter Estimation Methods

4.2.1 Estimation using System Identification Toolbox

In the System Identification Toolbox, identification is performed in the discrete time domain. For the identification of processes with direction-dependent responses, parametric estimation is used. This can be based on transfer-function or state-space models.

The general model structure of transfer function models [5, 45, 48] is

$$A(z)Y(z) = \frac{B(z)}{F(z)} z^{-nk}U(z) + \frac{C(z)}{D(z)} E(z) \tag{4.1}$$

where $A(z)$, $B(z)$, $C(z)$, $D(z)$ and $F(z)$ are polynomials of z, $Y(z)$ is the measured signal, nk is the delay (an integer value), $U(z)$ is the input signal and $E(z)$ is white noise.

Depending on the orders of the polynomials, different sets of models are available. The ones used in the identification of direction-dependent dynamics are the ARX and the ARMAX [49] models. For the ARX model (where AR refers to the autoregressive part $A(z)Y(z)$ and X to the extra input $B(z)U(z)$) the orders of the polynomials $C(z)$, $D(z)$ and $F(z)$ equal zero. Estimation is carried out using either least squares or instrumental variables method [50]. In the current application, the former was used because it produces results which are closer to the theoretical values for a first order process compared to the latter. For the ARMAX model (due to a moving average (MA) part $C(z)E(z)$) the orders of $D(z)$ and $F(z)$ are zero. Since no noise was present during the simulation, $C(z)$ was set to zero, making the model equivalent to the ARX model. Estimation was carried out using the prediction error method [51, 52]. The mean in both the input and output was removed before the start of the identification process.

The basic structure of a state-space model [45, 48, 49] is

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + Ke(t) \\ y(t) &= Cx(t) + Du(t) + e(t) \end{aligned} \tag{4.2}$$

where A, B, C, D and K are matrices, $u(t)$ and $y(t)$ are the input and output respectively, $x(t)$ is the state vector, and $e(t)$ is a stochastic process.

In the simulation, K was estimated. If K is fixed to zero, the model becomes equivalent to the Output-Error [45, 48] method. The matrix D can be fixed to zero or set as

variable. This will be further elaborated in Sections 4.3.1 and 4.3.2. Fixing D to zero means that there is a delay of at least one sample from the input to the output. The initial value of the state vector X_0 was estimated from the data during the simulation. (There was almost no change in the values of the estimated linear dynamics if X_0 was set to zero.)

The estimation method uses either the *pem* or the *n4sid* algorithms [45]. The *pem* is a standard prediction error method based on iterative minimisation of a criterion while *n4sid* is a subspace-based method that does not use iterative search. The *n4sid* method was used in the identification of systems with direction-dependent dynamics because with a first order process, it produces results which are closer to the theoretical values compared to the *pem* method.

4.2.2 Estimation using Frequency Domain System Identification Toolbox

In the Frequency Domain System Identification Toolbox, identification is performed in the frequency domain. The general model is given by

$$Y_{mk} = \delta \frac{N(\Omega_k)}{D(\Omega_k)} (U_{mk} - N_{uk}) + N_{yk} \tag{4.3}$$

where U_{mk} and Y_{mk} are the measured complex amplitudes at the input and output respectively, N_{uk} and N_{yk} are the measurement noises at the input and output respectively, and δ is a delay operator. $N(\Omega_k)$ and $D(\Omega_k)$ are polynomials of Ω_k, where $\Omega_k = s_k = j\omega_k$ in the s-domain, or $\Omega_k = z_k = \exp(j\omega_k T)$ in the z-domain [5].

The parametric model is estimated using the Estimator for Linear Systems (ELiS) [5] which is implemented using the function *elis*. The function *imppar* is used to read parameters from the parameter vectors. For the identification of direction-dependent

dynamics, the delay was set to zero and the data points at dc and the Nyquist frequency were deselected before the start of the identification process, which was carried out in the z-domain. (If the signal has even harmonics suppressed, the data points at these harmonics should also be removed since they carry no extra information.)

4.2.3 Estimation using Correlation Analysis

This is essentially a two-step method which first determines the correlation functions and then estimates the parameters of the parametric model using the method of least squares [46, 47]. The first step results in a nonparametric model. When a pseudo-random signal is used as input, the shape of the crosscorrelation function is approximately equal to the impulse response of the system due to the autocorrelation of the pseudo-random signal being approximately an impulse. The crosscorrelation function can be calculated using

$$R_{uy}(n) = \sum_{k=0}^{N-1} S_{uy}(k) \exp\left(\frac{2\pi kn}{N}\right) \tag{4.4}$$

and

$$S_{uy}(k) = U_k^* Y_k \tag{4.5}$$

where $S_{uy}(k)$ is the cross-power-spectral density between the input u and the output y, U_k^* is the complex conjugate of the discrete Fourier transform of the input, and Y_k is the discrete Fourier transform of the output.

Having obtained the approximate impulse response, the parameters a_i and b_i of the parametric model

$$\begin{aligned} y(k) = &-a_1 y(k-1) - a_2 y(k-2) - \ldots - a_m y(k-m) \\ &+ b_0 u(k-d) + b_1 u(k-d-1) + \ldots + b_{m-1} u(k-d-(m-1)) \end{aligned} \tag{4.6}$$

can be estimated using least squares method as shown in [46, 47].

Combining equations (4.4) to (4.6) in matrix form

$$g = Q\theta$$

where $g = \begin{bmatrix} R_{uy}(1) \\ R_{uy}(2) \\ R_{uy}(3) \\ .. \\ .. \\ .. \\ R_{uy}(l) \end{bmatrix}$, $\theta = \begin{bmatrix} a_1 \\ a_2 \\ .. \\ a_m \\ b_0 \\ b_1 \\ .. \\ b_{m-1} \end{bmatrix}$ and

$$Q = \begin{bmatrix} 0 & 0 & .. & 0 & 1 & 0 & .. & 0 \\ -R_{uy}(1) & 0 & .. & 0 & 0 & 1 & .. & 0 \\ -R_{uy}(2) & -R_{uy}(1) & .. & 0 & 0 & 0 & .. & 0 \\ .. & .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. & .. \\ -R_{uy}(l-1) & -R_{uy}(l-2) & .. & -R_{uy}(l-m) & 0 & 0 & .. & 0 \end{bmatrix} \quad (4.7)$$

l can be set equal to or less than the length of the input signal (or half its length if the signal is inverse-repeat).

4.3 Estimation of Linear Dynamics

4.3.1 Process with First Order Dynamics

When a pseudo-random signal is used to perturb a linear system, the shape of the crosscorrelation function is approximately equal to the impulse response of the system (since the autocorrelation function of the input signal has an approximate shape of an impulse at zero delay). In order to estimate the best linear model for the system in terms of a single combined time constant T_C, the rate of fall of the impulse response is compared with the rate of decrease of the input-output crosscorrelation function.

Equating these gives

$$e^{\frac{-t}{Tc}} = a^{\frac{t}{T}} \tag{4.8A}$$

where a is defined in (3.4B). Hence,

$$Tc = \frac{-T}{\ln a} \tag{4.8B}$$

The combined time constant is thus closer to the smaller time constant.

Results obtained for four different sets of values for T_U/T and T_D/T were compared with those produced using the ARX, ARMAX and state-space models in the System Identification Toolbox, ELiS in the Frequency Domain System Identification Toolbox, and the correlation analysis technique (with l in (4.7) set equal to the length of the input signal, or half the length if the signal was inverse-repeat). These are tabulated in Tables 4.1 and 4.2 respectively for a 127-digit MLB signal with feedback from stages 4 and 7, and its corresponding inverse-repeat signal. Theoretical results obtained using equation (4.8B) compare well with those produced using the above fitting algorithms. It can also be seen that more consistent results are produced using the inverse-repeat signal due to the effects of even order nonlinearities being eliminated (as was shown in Section 3.4.1.1).

T_U/T, T_D/T	T_C/T Theory	T_C/T ARX	T_C/T ARMAX	T_C/T State-space	T_C/T ELiS	T_C/T Correlation analysis
2, 4	2.72	2.67	2.67	2.78	2.75	2.67
1, 5	1.92	1.82	1.81	1.89	1.87	1.83
1, 7	2.07	1.94	1.94	2.05	2.03	1.98
1, 10	2.21	2.06	2.05	2.19	2.17	2.17

Table 4.1. Combined time constants for a first order process. The input signal is a 127-digit MLB signal with feedback from stages 4 and 7.

T_U/T, T_D/T	T_C/T Theory	T_C/T ARX	T_C/T ARMAX	T_C/T State-space	T_C/T ELiS	T_C/T Correlation analysis
2, 4	2.72	2.72	2.73	2.72	2.72	2.72
1, 5	1.92	1.90	1.91	1.89	1.91	1.91
1, 7	2.07	2.05	2.06	2.03	2.06	2.06
1, 10	2.21	2.18	2.19	2.14	2.19	2.19

Table 4.2. Combined time constants for a first order process. The input signal is a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7.

It is interesting to note that when matrix D in the state-space model was set as variable, the model produced had a zero and a pole. When D was fixed to zero, the model had only a pole and no zeros. The position of the pole was almost unchanged with different settings of D. The values given in Tables 4.1 and 4.2 were obtained setting D to zero.

The values of T_C found using correlation analysis varied slightly as l in (4.7) was varied. This was caused by the extra peaks in the crosscorrelation function due to second order and higher order terms distorting the shape of the approximate impulse response (as was discussed in Section 3.4.1.1). The peaks above a certain value of delay can be excluded by setting l to be less than or equal that particular value of delay. The variation in T_C with l is given in Table 4.3 for two different processes perturbed with an MLB signal with feedback from stages 4 and 7, and its corresponding inverse-repeat MLB signal. A similar set of results using an MLB signal with feedback from stages 1, 4, 6 and 7, and its corresponding inverse-repeat MLB signal is given in Table 4.4. More consistent results are again obtained using inverse-repeat signals as expected since even order peaks are no longer present in the crosscorrelation function. When such signals are used, there is a general decrease in the estimated value of T_C as l is increased. Unfortunately, the reason for this behaviour is not known.

l	$T_U = T$, $T_D = 5T$ MLB T_C/T	$T_U = T$, $T_D = 5T$ Inverse-repeat MLB T_C/T	$T_U = T$, $T_D = 10T$ MLB T_C/T	$T_U = T$, $T_D = 10T$ Inverse-repeat MLB T_C/T
10	1.85	1.93	2.19	2.22
30	1.85	1.92	2.19	2.21
50	1.86	1.92	2.19	2.21
70	1.85	1.92	2.17	2.21
90	1.85	1.92	2.18	2.20
110	1.83	1.91	2.17	2.20
127	1.83	1.91	2.17	2.19

Table 4.3. Combined time constants estimated using correlation analysis. The input is an MLB signal with feedback from stages 4 and 7, and its corresponding inverse-repeat signal.

l	$T_U = T$, $T_D = 5T$ MLB T_C/T	$T_U = T$, $T_D = 5T$ Inverse-repeat MLB T_C/T	$T_U = T$, $T_D = 10T$ MLB T_C/T	$T_U = T$, $T_D = 10T$ Inverse-repeat MLB T_C/T
10	1.84	1.93	2.15	2.23
30	1.83	1.93	2.15	2.22
50	1.84	1.93	2.15	2.22
70	1.82	1.92	2.14	2.21
90	1.73	1.92	1.97	2.21
110	1.80	1.91	2.13	2.20
127	1.81	1.91	2.12	2.20

Table 4.4. Combined time constants estimated using correlation analysis. The input is an MLB signal with feedback from stages 1, 4, 6 and 7, and its corresponding inverse-repeat signal.

It was found that state-space models have poorer fit in the frequency domain compared to the other models. Processes with direction-dependent dynamics are probably not so well described by state-space methods. An example is given in Figure 4.2 using a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7. The process has $T_U = T$ and $T_D = 5T$. The frequency response of the ARX model for the same set of data is shown in Figure 4.3. The frequency responses of the ARMAX and ELiS models are nearly identical to that of the ARX model. The actual frequency response of the process is given in Figure 4.1.

The shapes of the gain responses were compared with one another. The absolute values differ due to the different scaling used by each individual algorithm. (The gain is differently defined depending on the algorithm used.) The phase response of the state-space model continues to decrease due to a pure time delay which was set as variable and was estimated by the algorithm. (Setting the delay to zero produced a phase response similar to the actual phase response of the process. However, an extra zero was added to the model obtained.)

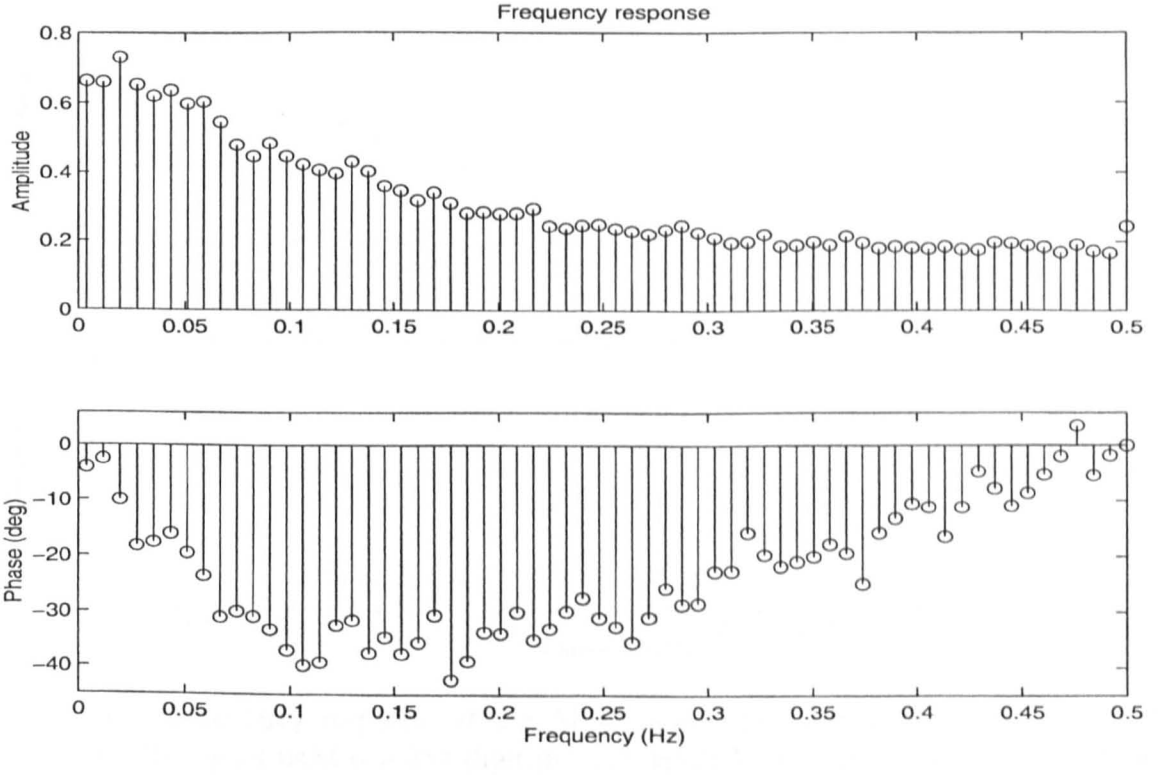

Figure 4.1. Frequency response of the process with $T_U = T$ and $T_D = 5T$.

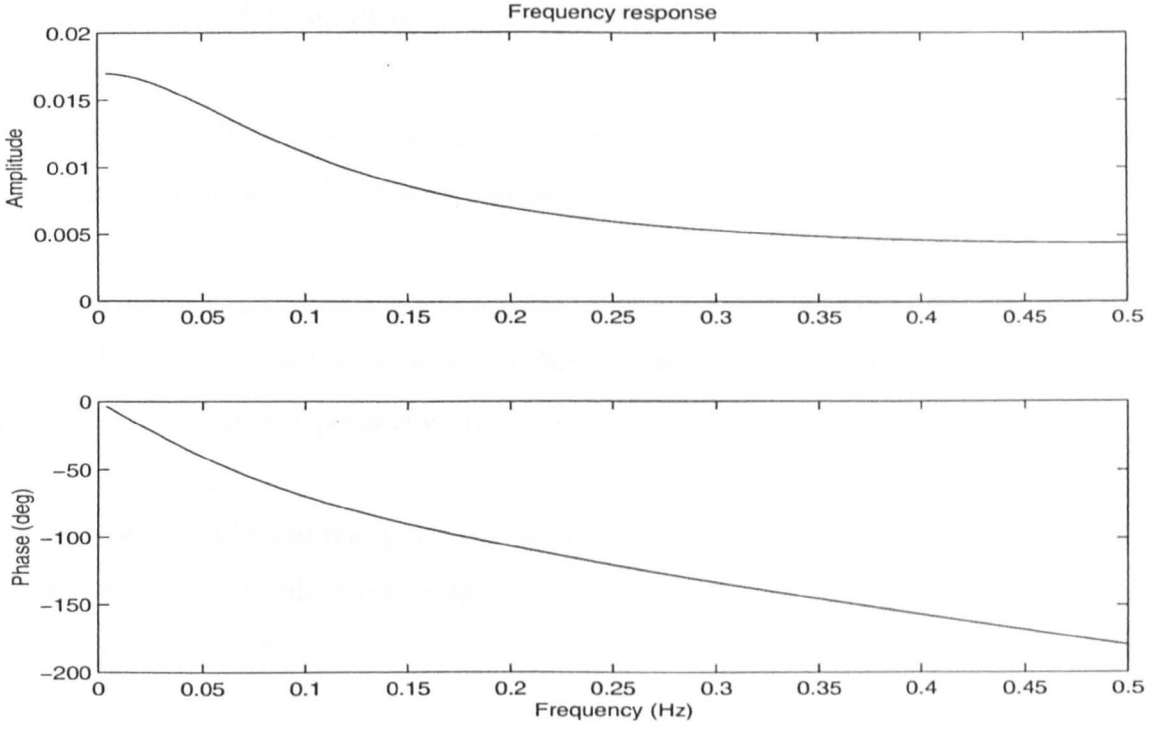

Figure 4.2. Frequency response of the state-space model for a process with $T_U = T$ and $T_D = 5T$. The signal used is a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7.

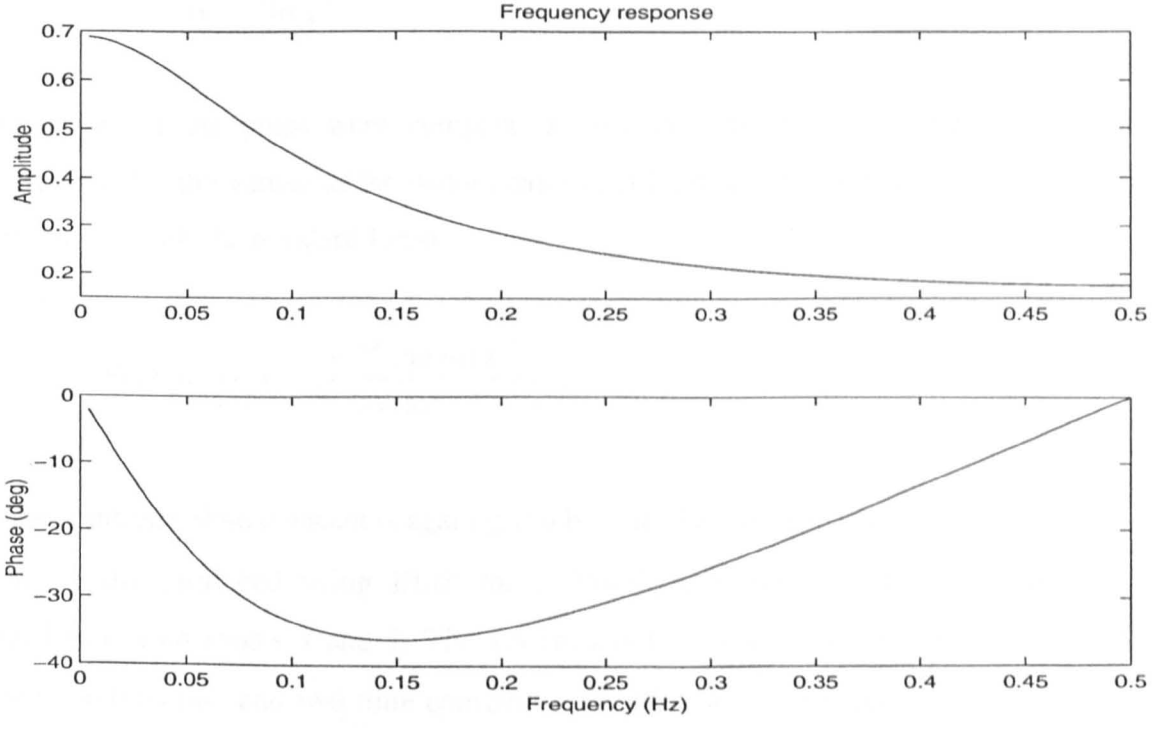

Figure 4.3. Frequency response of the ARX model for a process with $T_U = T$ and $T_D = 5T$. The signal used is a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7.

4.3.2 Process with Second Order Dynamics

For a process with second order dynamics, it is not possible to predict the combined time constants theoretically. Fitting algorithms are required for this purpose.

The process was modelled using two poles, with a zero added due to the presence of a zero order hold. The poles found were either real poles on the positive horizontal axis in the z-plane, or complex poles close to this axis.

If the poles found were real poles (on the positive horizontal axis), the combined time constants were each calculated using

$$z = \exp(-\alpha T) \tag{4.9A}$$

where z is the value of the pole being considered. This gives

$$T_C = \frac{1}{\alpha} = \frac{-T}{\ln z} \tag{4.9B}$$

However, if the poles were complex, a resonant frequency was then present. To calculate T_C, the values of the denominator coefficients of the z transfer function were compared with the standard formula

$$F(z) = \frac{e^{-\alpha T} \sin \omega T z^{-1}}{1 - 2e^{-\alpha T} \cos \omega T z^{-1} + e^{-2\alpha T} z^{-2}} \tag{4.10}$$

The combined time constant is again given by $1/\alpha$. An example is given in Figure 4.4 of the results produced using ELiS for a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7. The system has two time constants each of $5T$ in the positive direction and two time constants each of T in the negative direction ($\zeta = 1$ in both directions, $\omega_{nU} = 0.2/T$ and $\omega_{nD} = 1/T$). Using ELiS, the denominator of the z transfer function was found to be

$$0.5715 - 0.7750\, z^{-1} + 0.2666 z^{-2}$$

Comparing this with the denominator in the standard formula (equation (4.10)),

$$e^{-2\alpha T} = \frac{0.2666}{0.5715} = 0.4665$$

$\alpha = 0.381$ with $T = 1$ s as in the simulation. Hence,

$$T_C = \frac{1}{\alpha} = 2.62 \text{ s}$$

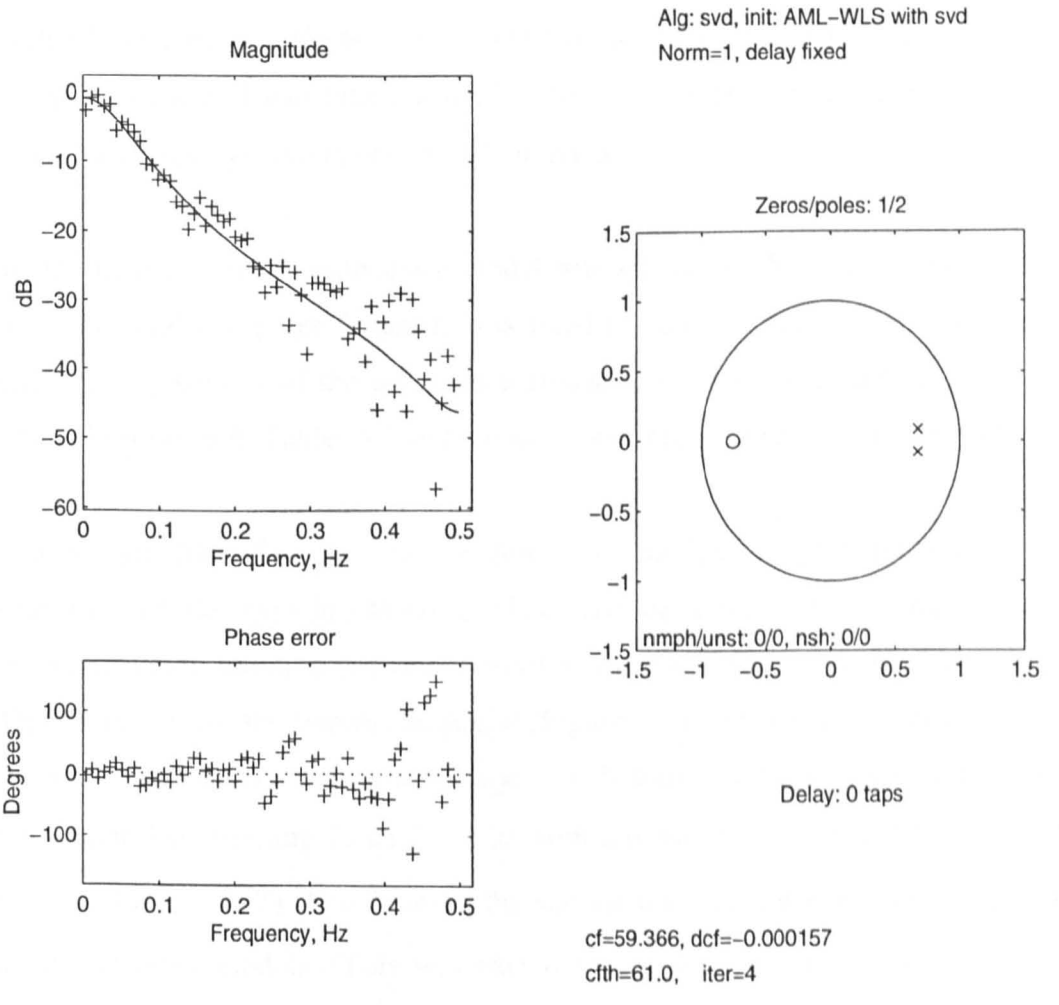

Figure 4.4. Results produced using ELiS for a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7. The system has $\zeta = 1$ in both directions, $\omega_{nU} = 0.2/T$ ($T_U = 5T$, $5T$) and $\omega_{nD} = 1/T$ ($T_D = T$, T). The solid line in the magnitude plot shows the gain response of the estimated model.

Results obtained for different combinations of process time constants using different fitting algorithms are given in Tables 4.5 and 4.6. In Table 4.5, the input is an inverse-repeat MLB signal with feedback from stages 4 and 7 while in Table 4.6, the input is an inverse-repeat MLB signal with feedback from stages 1, 4, 6 and 7. For two sets of faster dynamics with $\zeta_U = \zeta_D = 2$, the estimated values of damping factor (ζ_C) are not very close to the true value. Indeed, using ELiS, only a first order model could be estimated; when trying to estimate a second order model, one of the estimated time constants was very small, but negative.

The method of correlation analysis followed by least squares resulted in unstable poles, probably due to the effects of oversampling not being taken into account in equation (4.7). This method was thus not used in the estimation of linear dynamics for second order processes with direction-dependent dynamics.

When matrix D in the state-space model was set as variable, the model produced had two zeros and two poles. When D was fixed to zero, the model had one zero and two poles. The positions of the poles were almost unchanged with different settings of D. The values given in Tables 4.5 and 4.6 were obtained setting D to zero as before.

The System Identification Toolbox provides functions to plot the impulse and step responses of the models obtained. These can be compared with the shapes of the crosscorrelation function and the integral of the crosscorrelation function respectively. This is shown for the impulse response (Figure 4.5) and the step response (Figure 4.6) for a 254-digit inverse-repeat MLB signal with feedback from stages 1, 4, 6 and 7, and the system has damping factor $\zeta = 1$ in both directions, $\omega_{nU} = 0.2/T$ ($T_U = 5T$, $5T$) and $\omega_{nD} = 1/T$ ($T_D = T$, T). The fit using the state-space model was poorer compared to that using the other models. (This was true for most sets of simulations conducted.) The above could be linked to the poorer fit in the frequency domain. However, for a first order system, the discrepancy in the time domain was not obvious, for all sets of simulations conducted.

Process Dynamics	ARX	ARMAX	State-space	ELiS
$T_U / T = 0.05, 0.75$ $T_D / T = 0.21, 2.99$ $\zeta_U = \zeta_D = 2,$ $\omega_{nU} = 5/T, \omega_{nD} = 1.25/T$	$T_C / T = 0.64,$ 6.73 $\zeta_C = 1.78,$ $\omega_{nC} = 0.48/T$	$T_C / T = 0.65,$ 7.24 $\zeta_C = 1.82,$ $\omega_{nC} = 0.46/T$	$T_C / T = 1.30,$ 6.15 $\zeta_C = 1.32,$ $\omega_{nC} = 0.35/T$	$T_C / T = 1.28^*$
$T_U / T = 0.09, 1.24$ $T_D / T = 0.36, 4.98$ $\zeta_U = \zeta_D = 2,$ $\omega_{nU} = 3/T, \omega_{nD} = 0.75/T$	$T_C / T = 0.27,$ 1.90 $\zeta_C = 1.53,$ $\omega_{nC} = 1.41/T$	$T_C / T = 0.33,$ 1.70 $\zeta_C = 1.36,$ $\omega_{nC} = 1.34/T$	$T_C / T = 1.86,$ 4.31 $\zeta_C = 1.09,$ $\omega_{nC} = 0.35/T$	$T_C / T = 1.95^*$
$T_U / T = 0.27, 3.73$ $T_D / T = 1.07, 14.93$ $\zeta_U = \zeta_D = 2,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.25/T$	$T_C / T = 0.28,$ 6.73 $\zeta_C = 2.54,$ $\omega_{nC} = 0.73/T$	$T_C / T = 0.65,$ 7.24 $\zeta_C = 1.82,$ $\omega_{nC} = 0.46/T$	$T_C / T = 1.30,$ 6.15 $\zeta_C = 1.32,$ $\omega_{nC} = 0.35/T$	$T_C / T = 0.69,$ 6.84 $\zeta_C = 1.74,$ $\omega_{nC} = 0.46/T$
$T_U / T = 1, 1$ $T_D / T = 4, 4$ $\zeta_U = \zeta_D = 1,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.25/T$	$T_C / T = 2.13,$ 2.13 $\zeta_C = 0.98,$ $\omega_{nC} = 0.48/T$	$T_C / T = 1.60,$ 3.26 $\zeta_C = 1.06,$ $\omega_{nC} = 0.44/T$	$T_C / T = 3.01,$ 3.01 $\zeta_C = 0.82,$ $\omega_{nC} = 0.41/T$	$T_C / T = 2.59,$ 2.59 $\zeta_C = 0.89,$ $\omega_{nC} = 0.43/T$
$T_U / T = 1, 1$ $T_D / T = 5, 5$ $\zeta_U = \zeta_D = 1,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.2/T$	$T_C / T = 2.26,$ 2.26 $\zeta_C = 0.98,$ $\omega_{nC} = 0.45/T$	$T_C / T = 1.68,$ 3.88 $\zeta_C = 1.09,$ $\omega_{nC} = 0.39/T$	$T_C / T = 2.74,$ 2.74 $\zeta_C = 0.91,$ $\omega_{nC} = 0.40/T$	$T_C / T = 2.62,$ 2.62 $\zeta_C = 0.95,$ $\omega_{nC} = 0.40/T$
$T_U / T = 1 \pm j0.87$ $T_D / T = 2 \pm j3.46$ $\zeta_U = \zeta_D = 0.5,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.25/T$	$T_C / T = 3.44,$ 3.44 $\zeta_C = 0.68,$ $\omega_{nC} = 0.43/T$	$T_C / T = 3.58,$ 3.58 $\zeta_C = 0.70,$ $\omega_{nC} = 0.40/T$	$T_C / T = 4.97,$ 4.97 $\zeta_C = 0.56,$ $\omega_{nC} = 0.36/T$	$T_C / T = 4.56,$ 4.56 $\zeta_C = 0.59,$ $\omega_{nC} = 0.37/T$

Table 4.5. Combined time constants for second order processes estimated using different models/algorithms. The input is an inverse-repeat MLB signal with feedback from stages 4 and 7. *A second order model could not be estimated using ELiS in these two cases.

Process Dynamics	ARX	ARMAX	State-space	ELiS
$T_U / T = 0.05, 0.75$ $T_D / T = 0.21, 2.99$ $\zeta_U = \zeta_D = 2,$ $\omega_{nU} = 5/T, \omega_{nD} = 1.25/T$	$T_C / T = 0.53,$ 1.15 $\zeta_C = 1.08,$ $\omega_{nC} = 1.28/T$	$T_C / T = 0.53,$ 1.15 $\zeta_C = 1.08,$ $\omega_{nC} = 1.29/T$	$T_C / T = 1.25,$ 3.64 $\zeta_C = 1.15,$ $\omega_{nC} = 0.47/T$	$T_C / T = 1.47^*$
$T_U / T = 0.09, 1.24$ $T_D / T = 0.36, 4.98$ $\zeta_U = \zeta_D = 2,$ $\omega_{nU} = 3/T, \omega_{nD} = 0.75/T$	$T_C / T = 0.27,$ 1.91 $\zeta_C = 1.51,$ $\omega_{nC} = 1.40/T$	$T_C / T = 0.32,$ 1.71 $\zeta_C = 1.37,$ $\omega_{nC} = 1.34/T$	$T_C / T = 1.12,$ 2.02 $\zeta_C = 1.04,$ $\omega_{nC} = 0.67/T$	$T_C / T = 2.54^*$
$T_U / T = 0.27, 3.73$ $T_D / T = 1.07, 14.93$ $\zeta_U = \zeta_D = 2,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.25/T$	$T_C / T = 0.57,$ 7.36 $\zeta_C = 1.93,$ $\omega_{nC} = 0.49/T$	$T_C / T = 0.58,$ 7.69 $\zeta_C = 1.96,$ $\omega_{nC} = 0.47/T$	$T_C / T = 0.78,$ 7.06 $\zeta_C = 1.67,$ $\omega_{nC} = 0.43/T$	$T_C / T = 0.69,$ 7.02 $\zeta_C = 1.75,$ $\omega_{nC} = 0.45/T$
$T_U / T = 1, 1$ $T_D / T = 4, 4$ $\zeta_U = \zeta_D = 1,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.25/T$	$T_C / T = 1.53,$ 2.80 $\zeta_C = 1.05,$ $\omega_{nC} = 0.48/T$	$T_C / T = 1.39,$ 3.59 $\zeta_C = 1.11,$ $\omega_{nC} = 0.45/T$	$T_C / T = 2.37,$ 2.37 $\zeta_C = 0.96,$ $\omega_{nC} = 0.44/T$	$T_C / T = 2.37,$ 2.37 $\zeta_C = 0.96,$ $\omega_{nC} = 0.44/T$
$T_U / T = 1, 1$ $T_D / T = 5, 5$ $\zeta_U = \zeta_D = 1,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.2/T$	$T_C / T = 1.90,$ 2.98 $\zeta_C = 1.03,$ $\omega_{nC} = 0.42/T$	$T_C / T = 1.63,$ 4.31 $\zeta_C = 1.32,$ $\omega_{nC} = 0.38/T$	$T_C / T = 2.92,$ 2.92 $\zeta_C = 0.94,$ $\omega_{nC} = 0.37/T$	$T_C / T = 2.76,$ 2.76 $\zeta_C = 0.97,$ $\omega_{nC} = 0.37/T$
$T_U / T = 1 \pm j0.87$ $T_D / T = 2 \pm j3.46$ $\zeta_U = \zeta_D = 0.5,$ $\omega_{nU} = 1/T, \omega_{nD} = 0.25/T$	$T_C / T = 3.28,$ 3.28 $\zeta_C = 0.73,$ $\omega_{nC} = 0.41/T$	$T_C / T = 3.26,$ 3.26 $\zeta_C = 0.78,$ $\omega_{nC} = 0.40/T$	$T_C / T = 4.54,$ 4.54 $\zeta_C = 0.61,$ $\omega_{nC} = 0.36/T$	$T_C / T = 4.59,$ 4.59 $\zeta_C = 0.59,$ $\omega_{nC} = 0.37/T$

Table 4.6. Combined time constants for second order processes estimated using different models/algorithms. The input is an inverse-repeat MLB signal with feedback from stages 1, 4, 6 and 7. *A second order model could not be estimated using ELiS in these two cases.

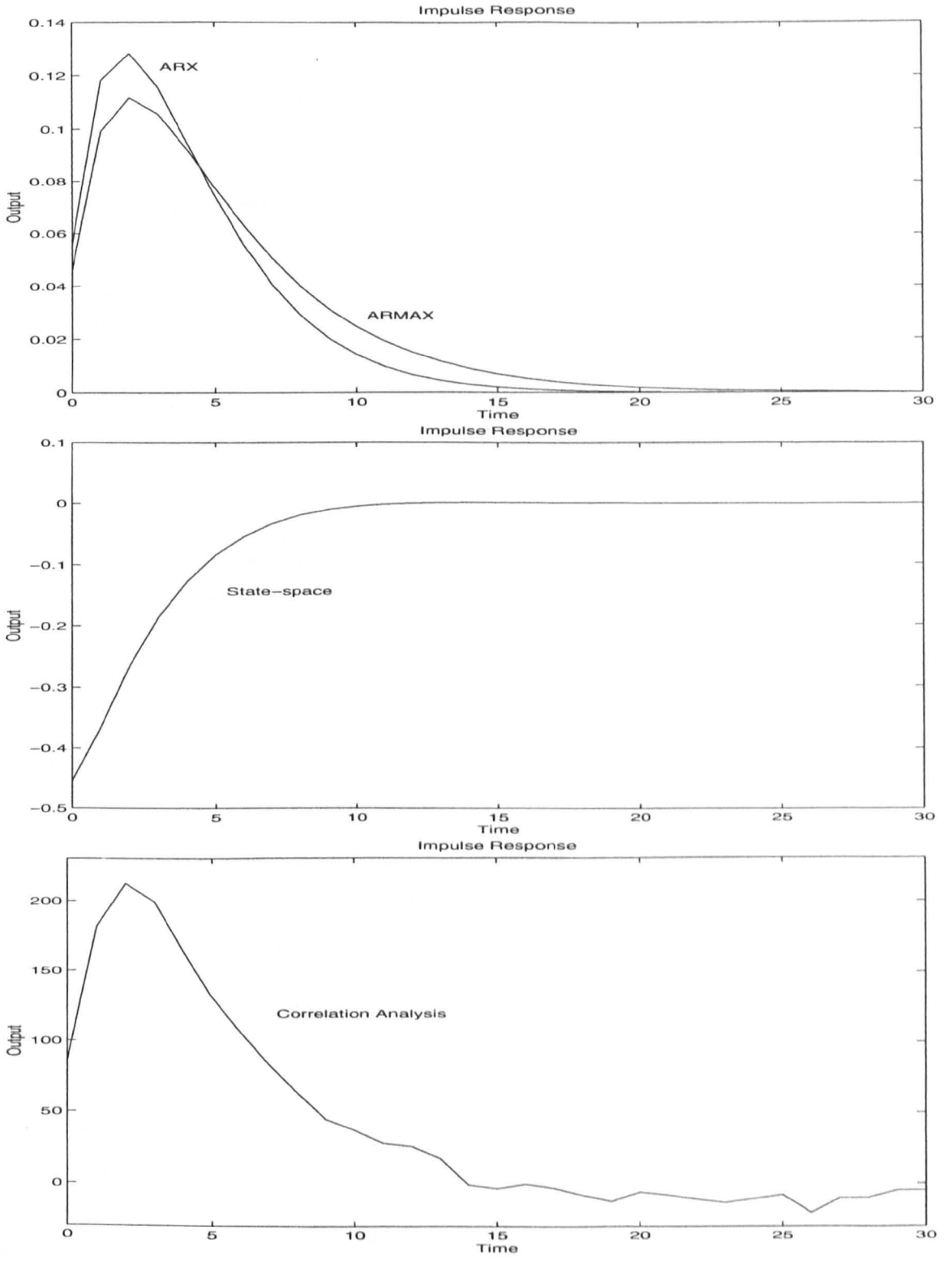

Figure 4.5. Impulse response of the ARX, ARMAX and state-space models, and that obtained through correlation analysis (the crosscorrelation function is plotted here since this approximates the shape of the impulse response) using a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7. The system has $\zeta = 1$ in both directions, $\omega_{nU} = 0.2/T$ ($T_U = 5T$, $5T$) and $\omega_{nD} = 1/T$ ($T_D = T$, T).

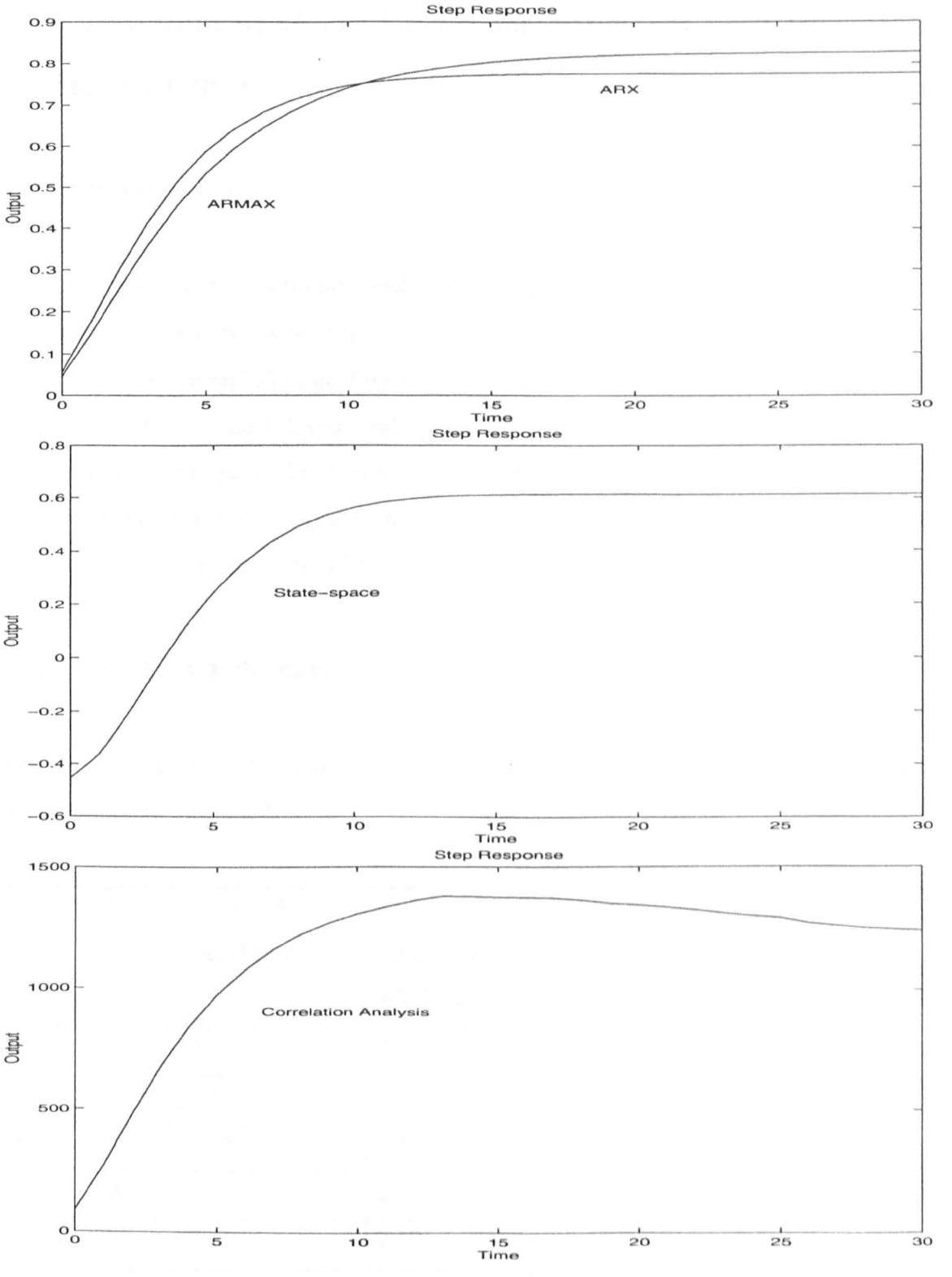

Figure 4.6. Step response of the ARX, ARMAX and state-space models, and that obtained through correlation analysis (the crosscorrelation function is plotted here since this approximates the shape of the impulse response) using a 254-digit inverse-repeat MLB signal with feedback from stages 4 and 7. The system has $\zeta = 1$ in both directions, $\omega_{nU} = 0.2/T$ ($T_U = 5T$, $5T$) and $\omega_{nD} = 1/T$ ($T_D = T$, T).

4.4 Estimation of Overall Gain for Processes with Direction-dependent Gains

4.4.1 Introduction

Some processes with direction-dependent dynamics have gains which are also direction-dependent. An example is the relationship between compressor speed and fuel flow for the Rolls-Royce Bristol Pegasus Engine [26] which is nonlinear, and about an operating point will have at least linear and quadratic nonlinear characteristics, resulting in direction-dependent gains. For a process in which the gain in the positive direction K_U is different from that in the negative direction K_D, the overall (combined) gain K_C of the system can be estimated using ELiS.

4.4.2 First Order Process

Results obtained for a few sets of first order processes with direction-dependent gains are tabulated in Table 4.7.

T_U/T	T_D/T	$K_U = 1, K_D = 4$ MLB input	$K_U = 1, K_D = 4$ Inverse-repeat MLB input	$K_U = 1, K_D = 6$ MLB input	$K_U = 1, K_D = 6$ Inverse-repeat MLB input
1	1	2.50	2.50	3.42	3.50
1	10	1.10	1.13	1.55	1.58
3	10	1.98	1.87	2.77	2.62
5	10	2.34	2.25	3.27	3.15
10	10	2.50	2.50	3.50	3.50

Table 4.7. Estimated overall gain K_C for first order processes with direction-dependent gains.

When the time constants in the positive and negative directions are equal ($T_U = T_D = T_C$), the estimated overall gain is approximately equal to the average of the gains in the two directions. Theoretically, K_C should be exactly equal to this value since from (3.1A) and (3.1B),

$$Y(s) = \frac{K_U}{sT_C + 1} U(s) \qquad \text{sgn}(\dot{y}) > 0 \qquad (4.11A)$$

$$Y(s) = \frac{K_D}{sT_C + 1} U(s) \qquad \text{sgn}(\dot{y}) < 0 \qquad (4.11B)$$

has the same effect in the output as a change in the signal levels.

If the input is binary, and assuming it has levels ±1 which is true for all the simulations in this chapter, the input signal can now be seen as having levels $+K_U$ and $-K_D$, while the process can be seen as having unity gain in both directions. There is therefore a bias of $(K_U - K_D)/2$ and an average signal amplitude of $(K_U + K_D)/2$. Hence, the effects of the nonlinearity in the system are not visible in this degenerate case, when using a binary perturbation signal.

However from Table 4.7, as the difference in the dynamics in the two directions increases, the estimated overall gain decreases and tends towards the smaller gain of the system. This is probably due to the fact that the output remains close to either the upper bound (which is determined by K_U) or the lower bound (which is determined by K_D) for most of the time during a period. An example is given in Figure 4.7 of the output signal using an MLB signal as the input. The process has gains $K_U = 1$ and $K_D = 6$. The time constants are $T_U = T$ and $T_D = 10T$. It can be seen from Figure 4.7 that the output signal stays close to the upper bound (which is 1 in this case) due to the faster time constant in the positive direction. However, any further increase is slow due to the signal being already close to the upper bound. Any decrease in the signal is also slow due to the larger time constant in the negative direction. It is possible that the overall gain decreased because a lot of the input signal power was lost in the process. The output using the same input for a process with gains $K_U = 1$, $K_D = 6$, $T_U = 10T$ and $T_D = 10T$ is given in Figure 4.8 for comparison. The output is seen to oscillate much more in

between the upper and lower bounds, and the overall gain is larger for this process. The use of an inverse-repeat signal is seen to yield more accurate results. Also, the values obtained for the estimated overall gain remained unchanged when the gains in the positive and negative directions were exchanged.

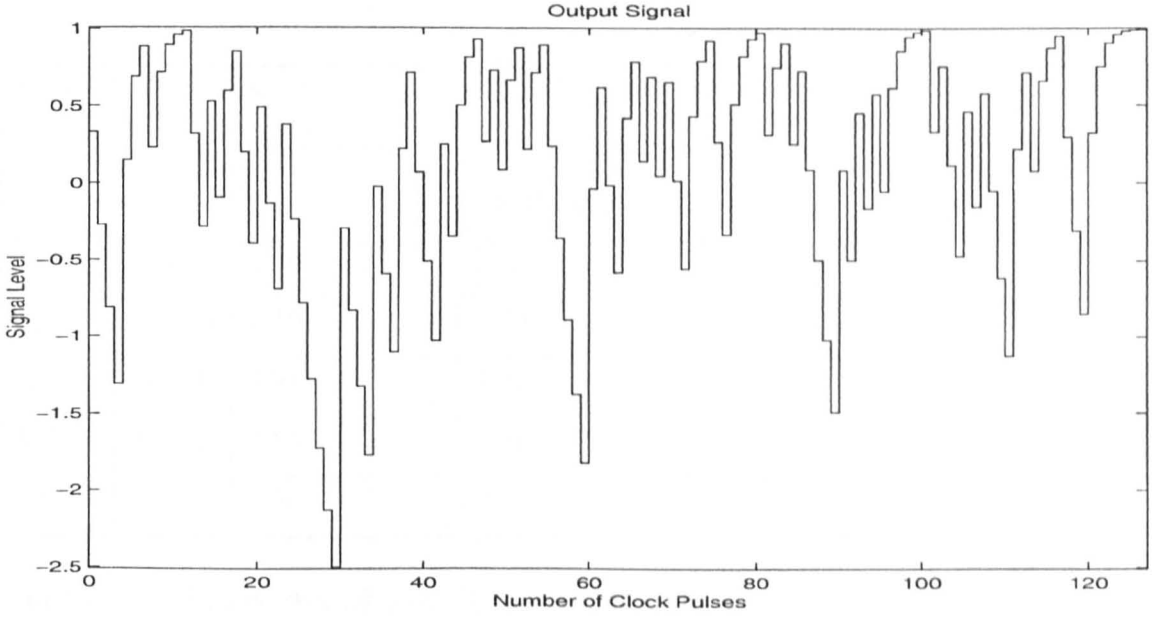

Figure 4.7. Output using an MLB signal for a process with $T_U = T$, $T_D = 10T$, $K_U = 1$ and $K_D = 6$. $K_C = 1.55$.

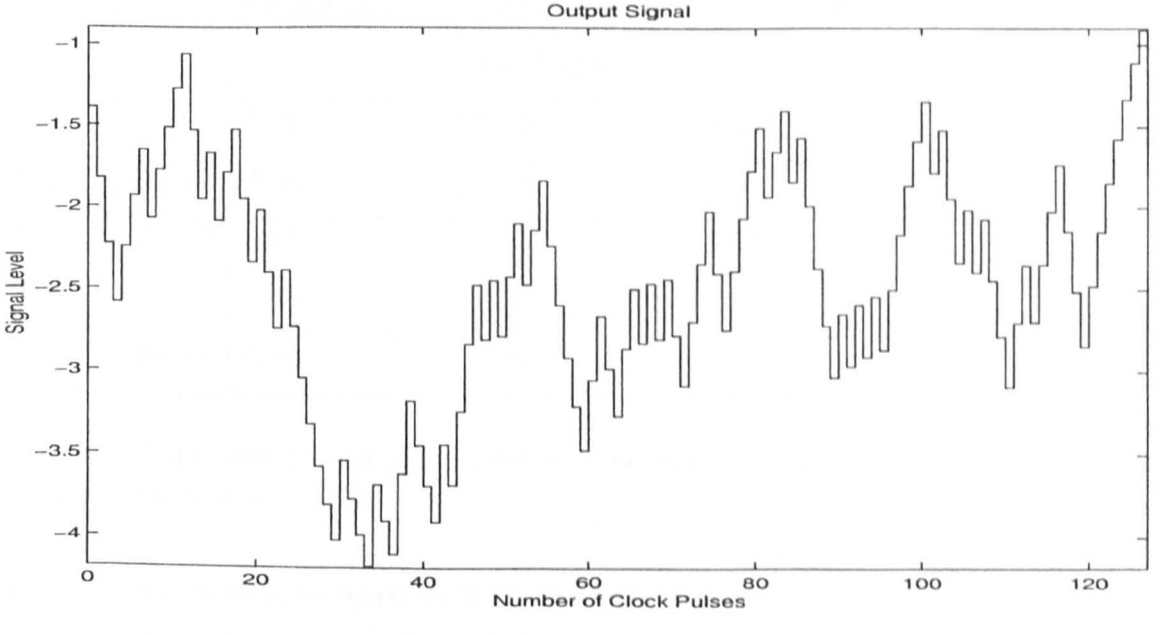

Figure 4.8. Output using an MLB signal for a process with $T_U = 10T$, $T_D = 10T$, $K_U = 1$ and $K_D = 6$. $K_C = 3.50$.

4.4.3 Second Order Process

Results obtained are tabulated in Table 4.8 for second order processes with larger gain in the positive direction ($K_U > K_D$), and in Table 4.9 for second order processes with larger gain in the negative direction ($K_U < K_D$).

T_U/T	T_D/T	$K_U = 4, K_D = 1$ MLB input	$K_U = 4, K_D = 1$ Inverse-repeat MLB input	$K_U = 6, K_D = 1$ MLB input	$K_U = 6, K_D = 1$ Inverse-repeat MLB input
1, 1	1, 1	2.50	2.50	3.50	3.50
1, 1	10, 10	2.19	1.35	5.74	1.89
3, 3	10, 10	2.91	2.42	4.06	3.39
5, 5	10, 10	2.55	2.46	3.57	3.45
10, 10	10, 10	2.50	2.50	3.49	3.50

Table 4.8. Estimated overall gain K_C for second order processes with larger gain in the positive direction ($K_U > K_D$).

T_U/T	T_D/T	$K_U = 1, K_D = 4$ MLB input	$K_U = 1, K_D = 4$ Inverse-repeat MLB input	$K_U = 1, K_D = 6$ MLB input	$K_U = 1, K_D = 6$ Inverse-repeat MLB input
1, 1	1, 1	2.50	2.50	3.50	3.50
1, 1	10, 10	4.43	1.35	6.20	2.85
3, 3	10, 10	2.91	2.42	4.08	3.39
5, 5	10, 10	2.55	2.46	3.57	3.45
10, 10	10, 10	2.50	2.50	3.50	3.50

Table 4.9. Estimated overall gain K_C for second order processes with larger gain in the negative direction ($K_U < K_D$).

As in the first order case, when the time constants in the positive and negative directions are equal, the estimated overall gain is approximately equal to the average of the gains in the upward and downward directions, which is to be expected theoretically. However,

as the difference between the dynamics in the two directions increases, the overall gain deviates from this average value. This cannot be easily explained because the output does not respond immediately to a change in input. Examples are given in Figures 4.9 and 4.10 of the output using an inverse-repeat MLB signal as the input. The process has a gain of 6 in the positive direction and 1 in the negative direction. In Figure 4.9, the process has $\zeta = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.1/T$ ($T_D = 10T$, $10T$) while in Figure 4.10, the process has $\zeta = 1$ and $\omega_{nU} = \omega_{nD} = 0.1/T$ ($T_U = T_D = 10T$, $10T$). The estimated overall gain is no longer independent of the direction in which the process has a larger gain. However, it was found that there is a higher probability of obtaining the same overall gain when the gains in the two directions are exchanged, if an inverse-repeat MLB signal is used.

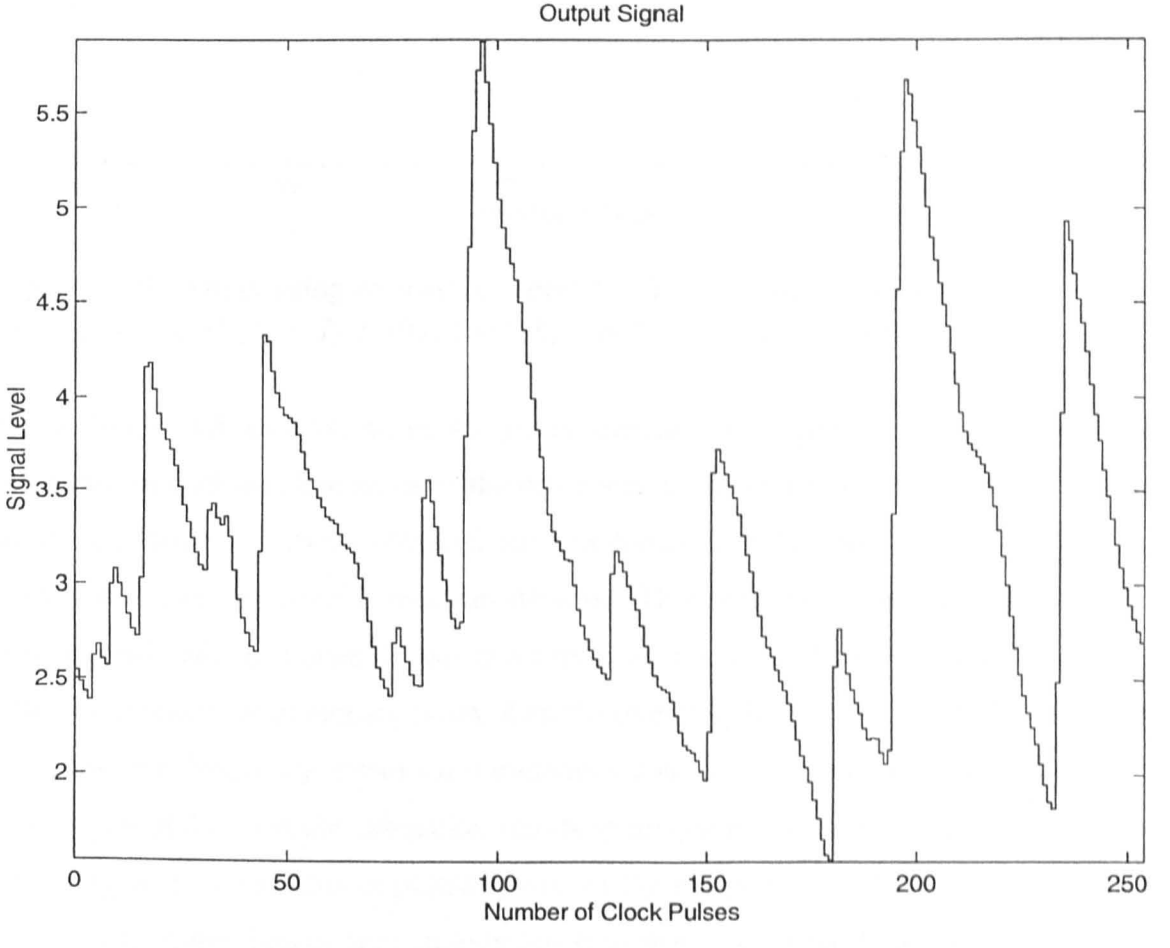

Figure 4.9. Output using an inverse-repeat MLB signal for a process with $\zeta = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.1/T$ ($T_D = 10T$, $10T$). $K_U = 6$, $K_D = 1$ and $K_C = 1.89$.

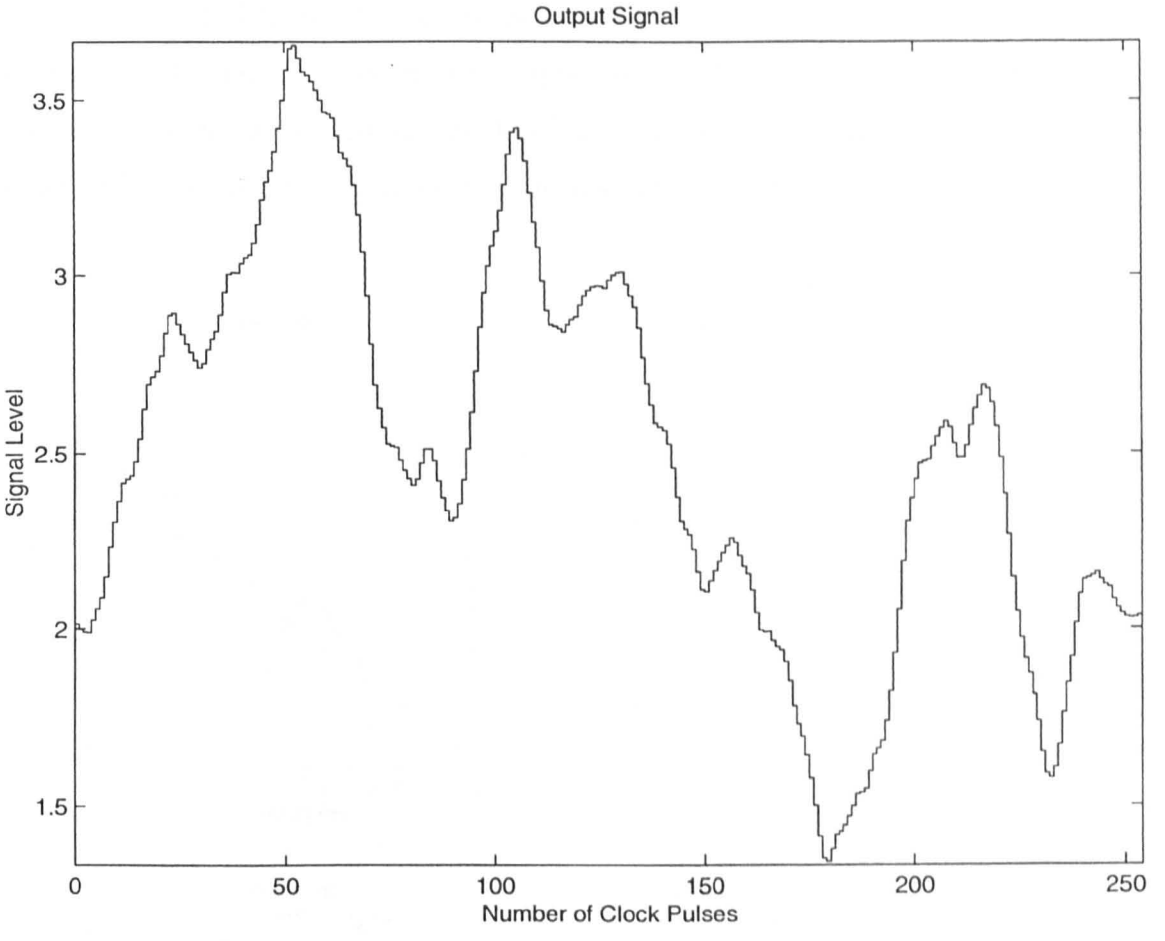

Figure 4.10. Output using an inverse-repeat MLB signal for a process with $\zeta = 1$ and $\omega_{nU} = \omega_{nD} = 0.1/T$ ($T_U = T_D = 10T$, $10T$). $K_U = 6$, $K_D = 1$ and $K_C = 3.50$.

From Tables 4.8 and 4.9, when an inverse-repeat MLB signal is used as input, the estimated overall gain decreases as the difference in the dynamics in the two directions increases. However, results obtained are less consistent when an MLB signal is used. The overall gain can either increase or decrease. This depends on the fitting using ELiS. The overall gain is higher if the estimation results in real poles. However, if the estimation results in imaginary poles, then the overall gain is lower due to the presence of a resonant frequency. From the simulations conducted, it was found that there is a higher probability that the estimation results in imaginary poles when an inverse-repeat MLB signal is used. This is probably due to the magnitude of the second frequency point being larger than or very slightly less than that of the first. Examples are given in Figures 4.11 and 4.12 for a process with $\zeta = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.1/T$

($T_D = 10T$, $10T$). The gain is 1 in the positive direction and 4 in the negative direction. In Figure 4.11, the input is an MLB signal while in Figure 4.12, the corresponding inverse-repeat signal is used. It was also found that the estimated overall gain does not necessarily fall between the gains in the positive and negative directions.

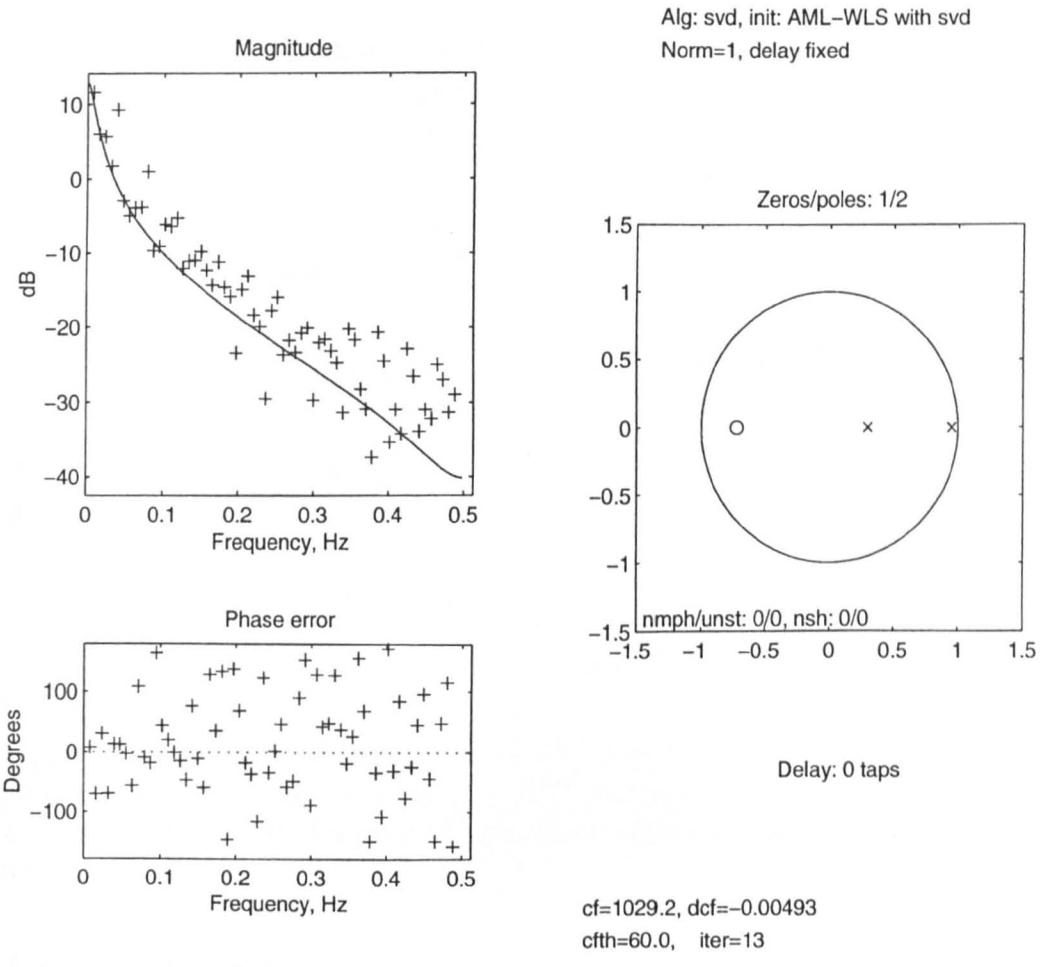

Figure 4.11. Results obtained from ELiS using an MLB signal for a process with $\zeta = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.1/T$ ($T_D = 10T$, $10T$). $K_U = 1$, $K_D = 4$ and $K_C = 4.43$. The solid line in the magnitude plot shows the gain response of the estimated model.

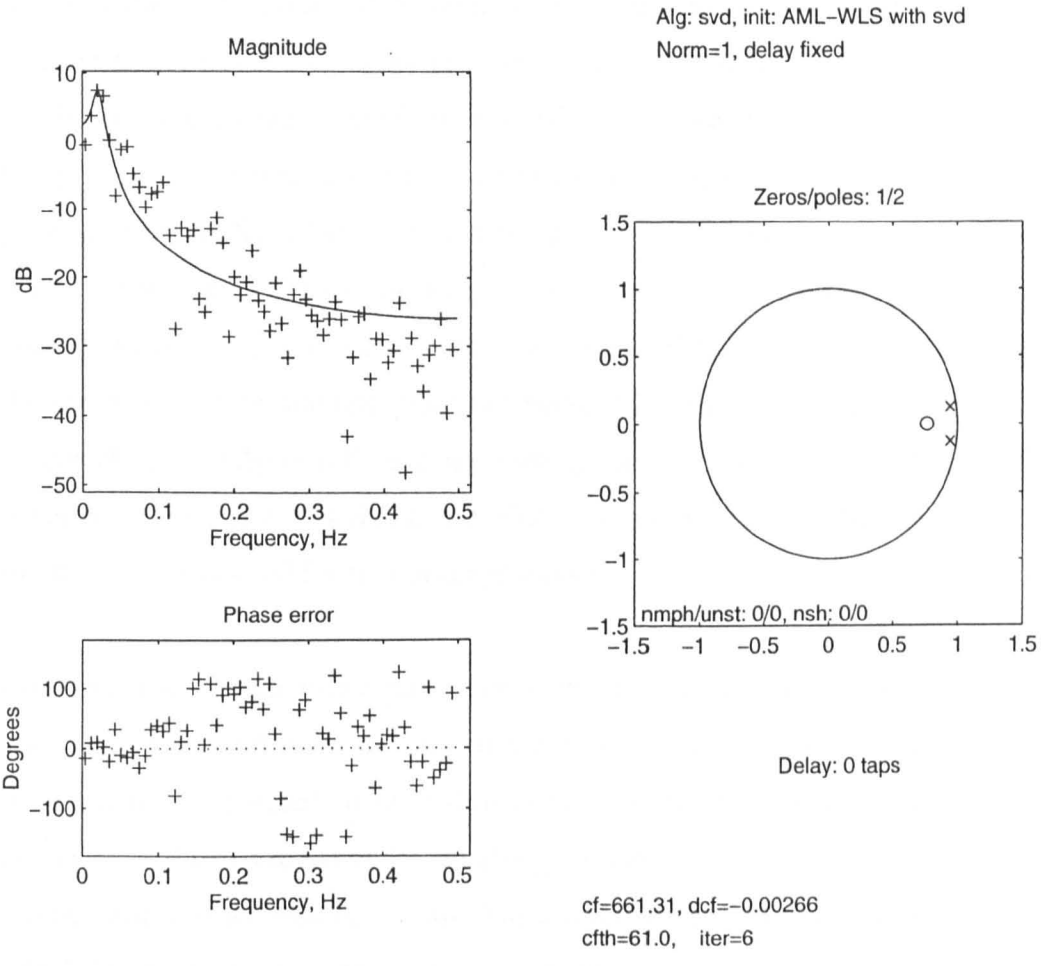

Figure 4.12. Results obtained from ELiS using an inverse-repeat MLB signal for a process with $\zeta = 1$, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.1/T$ ($T_D = 10T$, $10T$). $K_U = 1$, $K_D = 4$ and $K_C = 1.35$. The solid line in the magnitude plot shows the gain response of the estimated model.

4.5 Conclusions

In most practical situations, linear models are greatly preferred to their nonlinear counterparts due to their inherent simplicity. In light of this, it is not difficult to comprehend why nonlinear processes are often linearised to allow further analysis and simulation. In this chapter, the combined linear dynamics for a process with direction-dependent dynamics were investigated. Despite the process having linear dynamics in both directions, a departure from linearity was observed, due to the direction-dependent behaviour. In order to obtain a high accuracy and consistency in the estimation results,

an inverse-repeat signal should be used since this eliminates the effects of even order nonlinearities. For a first order process, simple calculations can be used to obtain the combined time constant, which was found to be closer to the smaller time constant of the process. However, theoretical results could not be obtained for a second order process. The ARX, ARMAX and state-space models, and the ELiS algorithm, can be used to estimate the linear model. It was found that the models obtained using state-space have poorer fit compared with those of the other models mentioned above. Hence, the transfer function models should be preferred for this application. Also, the method of correlation analysis followed by least squares results in unstable poles for second order processes, probably due to the effects of oversampling. Thus, in this chapter, this method was only used for first order processes.

For first and second order processes with direction-dependent gains, the theoretical overall gain is equal to the average of the gains in the positive and negative directions if the dynamics are equal in both directions. For the first order case, the overall gain decreases and tends towards the smaller gain with an increase in the difference between the dynamics in the two directions. The overall gain is independent of the direction in which the process has a larger gain. Unfortunately, results obtained for second order processes were not so conclusive. The overall gain depends on several factors, including the input perturbation signal applied, the dynamics of the particular process and the direction in which the process has a larger gain.

Chapter 5

The Use of Wiener Models to Describe Systems with Direction-dependent Dynamics

5.1 Abstract

The identification of systems with direction-dependent dynamics is extended to those in which the gain is also direction-dependent. The terms in the input-output crosscorrelation function are developed for a first order process perturbed with an MLB signal and an inverse-repeat MLB signal. It is shown that coherent patterns in the crosscorrelation function of a Wiener process can be made very similar to that of a direction-dependent process, provided that the model parameters are appropriately chosen. This can be achieved using the MATLAB Optimization Toolbox. Simulation results are used to verify the theoretical analysis. The linear dynamics for both direction-dependent and Wiener processes are estimated and compared. For a second order process, analytical results could not be obtained and simulation results are presented instead.

5.2 Introduction

In Chapter 3, the detection of processes with direction-dependent dynamics was discussed. It is shown in [25, 26] that pseudo-random binary signals based on maximum length sequences are the most suitable signals for the detection of such a departure from linearity. When these signals are used to perturb a direction-dependent process, coherent patterns are observed in the input-output crosscorrelation function that are not present when other kinds of pseudo-random binary signals are used. In this chapter, the theory is extended to the more general case in which the gain is also direction-dependent. Such processes may be modelled by one of the many derivatives of the Volterra functional series [28]. In this chapter, the modelling of the direction-dependent system using one of these derivatives, namely a Wiener process [11, 53 - 60], is investigated. This process has a structure in which a dynamic linear subsystem precedes an instantaneous nonlinearity. The parameters of the Wiener model are obtained using the MATLAB Optimization Toolbox [61].

The linear dynamics for both direction-dependent and Wiener processes are then estimated using the ARX and the ARMAX models in the System Identification Toolbox [45], ELiS in the Frequency Domain System Identification Toolbox [5] and correlation analysis with least squares [46, 47]. Different optimisation criteria are used for the Wiener models, and the results obtained are compared.

Unlike in the first order case, theoretical results could not be obtained for a second order process and simulation results are presented instead. It will be shown that the match between the direction-dependent and the Wiener processes is not as good as in the former case, except when the direction-dependent process has a large damping factor and fast dynamics, and therefore behaves like a first order process. In such cases, a Wiener model with first order dynamics can be used to model the actual process.

5.3 First Order Direction-dependent Process

5.3.1 Derivation of Process Output

For a first order process with direction-dependent dynamics and direction-dependent gains, the response y to an input u with levels ±1 is given by

$$y_t = K_U u + (y_{t-1} - K_U u)\exp(-T / T_U) \qquad u = +1 \qquad (5.1A)$$

$$y_t = K_D u + (y_{t-1} - K_D u)\exp(-T / T_D) \qquad u = -1 \qquad (5.1B)$$

where K_U, K_D and T are the gains in the positive and negative directions, and the clock-pulse interval respectively.

This can be described by the difference equation

$$y_t = \kappa_t y_{t-1} + (1-\kappa_t)(Au_t + F) \qquad (5.2)$$

where κ_t is defined in equation (3.4A).

$$A = \frac{K_U + K_D}{2} \qquad \text{and} \qquad (5.3A)$$

$$F = \frac{K_U - K_D}{2} \qquad (5.3B)$$

Substituting for κ_t we obtain

$$\begin{aligned} y_t &= (a+bu_t)y_{t-1} + (1-a-bu_t)(Au_t + F) \\ &= ay_{t-1} + bu_t y_{t-1} + (A - aA - bF)u_t - bA + (1-a)F \\ &= (a+bu_t)\left(y_{t-1} + \left(\frac{A-aA-bF}{b}\right)\right) - bA + (1-a)F - \left(\frac{A-aA-bF}{b}\right)a \end{aligned} \tag{5.4}$$

On rearranging,

$$\left(y_t + \frac{A-aA-bF}{b}\right) = (a+bu_t)\left(y_{t-1} + \frac{A-aA-bF}{b}\right) - A\left(b - \frac{(1-a)^2}{b}\right) \tag{5.5}$$

which is of the form $\alpha_t = \beta_t \alpha_{t-1} + \gamma$, where $\alpha_t = \left(y_t + \frac{A-aA-bF}{b}\right)$, $\beta_t = a + bu_t$ and $\gamma = -A\left(b - \frac{(1-a)^2}{b}\right)$.

Taking corresponding expressions for α_t, α_{t-1}, α_{t-2}, . . . , and multiplying them by β_t, $\beta_t\beta_{t-1}$, $\beta_t\beta_{t-1}\beta_{t-2}$, . . . , as in [25],

$$\alpha_t = \prod_{l=0}^{v} \beta_{t-l}\alpha_{t-v-1} + \sum_{p=0}^{v}\prod_{l=0}^{p} \gamma\beta_{t-l} \tag{5.6}$$

For stable processes, $a < 1$ and $b < 1$, so that as v becomes large, $\prod_{l=0}^{v}(a+bu_{t-l})$ approaches zero [25] leaving only the second term on the right-hand side of (5.6). Hence from (5.5), y_t can be expressed as a nonlinear function of the inputs u_{t-l}.

$$\begin{aligned} y_t &= A\left(\frac{(1-a)^2}{b} - b\right)\sum_{p=0}^{v}\prod_{l=0}^{p}(a+bu_{t-l}) - \frac{A-aA-bF}{b} \\ &= A\left(\left(\frac{(1-a)^2}{b} - b\right)\sum_{p=0}^{v}\prod_{l=0}^{p}(a+bu_{t-l}) - \left(\frac{1-a}{b}\right)\right) + F \end{aligned} \tag{5.7}$$

The right-hand side of (5.7) can be split into a constant term, a first order term and higher order terms [25, 26] as was done in Section 3.4.1.1.

Constant term

$$Y_0 = A\left(K(1+a+a^2+\ldots) - \frac{1-a}{b} \right) + F = -\frac{Ab}{1-a} + F \tag{5.8}$$

where $K = \frac{(1-a)^2}{b} - b$ as defined in equation (3.6).

First order term

$$Y_1 = AK\frac{b}{1-a}(u_t + au_{t-1} + a^2u_{t-2} + a^3u_{t-3} + a^4u_{t-4} + \ldots) \tag{5.9}$$

Second order terms

$$\begin{aligned} Y_2 = AK\frac{b^2}{1-a}(&u_tu_{t-1} + au_{t-1}u_{t-2} + a^2u_{t-2}u_{t-3} + a^3u_{t-3}u_{t-4} + \ldots \\ &+ au_tu_{t-2} + a^2u_{t-1}u_{t-3} + a^3u_{t-2}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-3} + a^3u_{t-1}u_{t-4} + \ldots \\ &+ a^3u_tu_{t-4} + \ldots \\ &+ \ldots) \end{aligned} \tag{5.10}$$

Third order terms

$$\begin{aligned} Y_3 = AK\frac{b^3}{1-a}(&u_tu_{t-1}u_{t-2} + au_{t-1}u_{t-2}u_{t-3} + a^2u_{t-2}u_{t-3}u_{t-4} + \ldots \\ &+ au_tu_{t-1}u_{t-3} + a^2u_{t-1}u_{t-2}u_{t-4} + \ldots \\ &+ au_tu_{t-2}u_{t-3} + a^2u_{t-1}u_{t-3}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-1}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-2}u_{t-4} + \ldots \\ &+ a^2u_tu_{t-3}u_{t-4} + \ldots \\ &+ \ldots) \end{aligned} \tag{5.11}$$

Since, in practice, $|b|$ is considerably less than unity, fourth order and higher order terms will be very small and can be neglected.

5.3.2 Derivation of Crosscorrelation Function using Maximum Length Binary Signals

If the input u is a pseudo-random binary signal of any class, with levels ±1, the autocorrelation function is +1 on-peak (at zero delay) and -1/N off-peak. Suppose that there are (N+1)/2 clock-pulse intervals at signal level +1 and (N-1)/2 clock-pulse intervals at signal level -1; then the contributions of Y_0 and Y_1 to the crosscorrelation function are :

Contribution of Y_0

$$W_0 = \left(-\frac{Ab}{1-a} + F\right)\left(\frac{1}{N}\right) \tag{5.12}$$

since the average value of the input signal is 1/N.

Contribution of Y_1

This contributes to a constant and a first order term in the crosscorrelation function. The on-peak value of the autocorrelation of the input contributes to a term

$$W_{1A}(iT) = \frac{Ac}{1-a}a^i \qquad i = 0, 1, 2, \ldots \tag{5.13}$$

where $c = (1-\exp(-T/T_U))(1-\exp(-T/T_D)) = Kb$ (5.14)

However, the off-peak value of the autocorrelation has a value of -1/N. This contributes to a constant and a first order term.

$$W_{1B}(iT) = \frac{Ac}{1-a}\left[(1+a+a^2+a^3+\ldots) - a^i\right]\left(\frac{-1}{N}\right) \qquad i = 0, 1, 2, \ldots \tag{5.15}$$

Thus the total contribution of Y_1 is

$$W_1(iT) = \frac{Ac}{1-a}\left[a^i\left(1+\frac{1}{N}\right) + \left[1+a+a^2+a^3+\ldots\right]\left(\frac{-1}{N}\right)\right]$$

$$= \frac{Ac}{1-a}\left[a^i\left(1+\frac{1}{N}\right) - \left(\frac{1}{N(1-a)}\right)\right] \tag{5.16}$$

Suppose now that the input is a pseudo-random MLB signal so that the shift-and-add property

$$s_i \oplus_2 s_{i-r} = s_{i-i_r} \qquad r = 1, 2, 3, \ldots \qquad (5.17)$$

applies. Here, $s(i)$ is a maximum length binary sequence with characteristic polynomial $f(D)$. $u(t)$ is obtained by converting the zero elements of $s(i)$ to -1 and the unity elements to +1. The values of i_r can be determined by dividing the polynomial $1 \oplus_2 D^r$ by the characteristic polynomial $f(D)$ until the single-term remainders D^{i_r} are obtained [62].

Contribution of Y_2

The first, second and third lines of (5.10) contribute to the terms

$$W_{2A}(iT) = \frac{-Acb}{1-a}\left[a^{i-i_1}\left(1+\frac{1}{N}\right)-\left(\frac{1}{N(1-a)}\right)\right] \qquad i = i_1, i_1+1, i_1+2, \ldots \qquad (5.18)$$

$$W_{2B}(iT) = \frac{-Acb}{1-a}\left[a^{1+i-i_2}\left(1+\frac{1}{N}\right)-\left(\frac{a}{N(1-a)}\right)\right] \qquad i = i_2, i_2+1, i_2+2, \ldots \qquad (5.19)$$

$$W_{2C}(iT) = \frac{-Acb}{1-a}\left[a^{2+i-i_3}\left(1+\frac{1}{N}\right)-\left(\frac{a^2}{N(1-a)}\right)\right] \qquad i = i_3, i_3+1, i_3+2, \ldots \qquad (5.20)$$

respectively. Thus the total contribution of Y_2 is

$$W_2(iT) = \frac{-Acb}{1-a}\left[\sum_{r=1}^{N-1} a^{r-1+i-i_r}\left(1+\frac{1}{N}\right)-\left(\frac{1}{N(1-a)^2}\right)\right] \qquad i = i_r, i_r+1, i_r+2, \ldots \qquad (5.21)$$

For an MLB signal, the following property also applies

$$s_i \oplus_2 s_{i-q} \oplus_2 s_{i-q-r} = s_{i-i_{q,r}} \qquad q = 1, 2, 3, \ldots \text{ and } r = 1, 2, 3, \ldots \qquad (5.22)$$

The values of $i_{q,r}$ can be determined by dividing the polynomial $1 \oplus_2 D^q \oplus_2 D^{q+r}$ by the characteristic polynomial $f(D)$ until the single-term remainders $D^{i_{q,r}}$ are obtained [62].

Contribution of Y_3

Similarly, this contributes to a constant and to third order terms.

$$W_3(iT) = \frac{Acb^2}{1-a}\left[\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a^{q+r-2+i-i_{q,r}}\left(1+\frac{1}{N}\right)-\left(\frac{1}{N(1-a)^3}\right)\right]$$

$$i = i_{q,r}, i_{q,r}+1, i_{q,r}+2, \ldots \qquad (5.23)$$

Grouping all the constant terms (including contributions from higher order terms), we obtain

$$\Phi_0 = \left(-\frac{Ab}{1-a}+F\right)\left(\frac{1}{N}\right)-\frac{Ac}{N(1-a)(1-a+b)}=\frac{-A+F}{N}=\frac{-K_D}{N} \tag{5.24}$$

Note that K_U does not enter into the expression for Φ_0; K_U affects the peak values relative to this constant offset.

Also the linear, quadratic and cubic terms in the crosscorrelation function are

$$\Phi_1(iT) = \frac{Ac}{1-a}a^i\left(1+\frac{1}{N}\right) \qquad i = 0, 1, 2, \ldots \tag{5.25}$$

$$\Phi_2(iT) = \frac{-Acb}{1-a}\sum_{r=1}^{N-1} a^{r-1+i-i_r}\left(1+\frac{1}{N}\right) \qquad i = i_r, i_r+1, i_r+2, \ldots \tag{5.26}$$

$$\Phi_3(iT) = \frac{Acb^2}{1-a}\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a^{q+r-2+i-i_{q,r}}\left(1+\frac{1}{N}\right) \qquad i = i_{q,r}, i_{q,r}+1, i_{q,r}+2, \ldots \tag{5.27}$$

respectively.

When $T_U = T_D$ (so that $b = 0$), but $K_U \neq K_D$, additional peaks no longer appear in the crosscorrelation function, due to the fact that the signal is binary. In this respect, this signal is not persistently exciting (see Section 4.4.2).

5.3.3 Derivation of Crosscorrelation Function using Inverse-repeat Maximum Length Binary Signals

Suppose now that the input u is an inverse-repeat signal with levels ± 1, obtained by inverting every other digit in a pseudo-random binary signal (of any class) of length N [9]. The resulting signal is of length $2N$ and has an autocorrelation function $R_{uu}(i)$ given by +1 for $i = 0 \bmod 2N$; -1 for $i = N \bmod 2N$; $+1/N$ for $i = 1, 3, 5, \ldots \neq N \bmod 2N$; and $-1/N$ for $i = 2, 4, 6, \ldots \neq 0 \bmod 2N$. For such a signal, the terms in the crosscorrelation function due to Y_0, Y_1 and Y_2 are :

Contribution of Y_0

$$W_0 = 0 \tag{5.28}$$

since the average value of the input signal is zero.

Contribution of Y_1

Considering only the first half of the crosscorrelation function (as the second half is the negative of the first), this contributes to a first order term. The on-peak value of the autocorrelation of the input contributes to a term

$$W_{1A}(iT) = \frac{Ac}{1-a} a^i \qquad i = 0, 1, 2, \ldots \tag{5.29}$$

However, the off-peak value of the autocorrelation as given above contributes to

$$W_{1B}(iT) = \frac{Ac}{1-a}\left[\left[-1+a-a^2+a^3-\ldots\right]\left(\frac{(-1)^i}{N}\right)+a^i\left(\frac{1}{N}\right)\right] \quad i = 0, 1, 2, \ldots \tag{5.30}$$

Thus the total contribution of Y_1 is

$$\begin{aligned} W_1(iT) &= \frac{Ac}{1-a}\left[a^i\left(1+\frac{1}{N}\right)+\left[-1+a-a^2+a^3-\ldots\right]\left(\frac{(-1)^i}{N}\right)\right] \\ &= \frac{Ac}{1-a}\left[a^i\left(1+\frac{1}{N}\right)-\left(\frac{(-1)^i}{N(1+a)}\right)\right] \end{aligned} \tag{5.31}$$

Contribution of Y_2

Due to the inverse-repeat nature of the signal,

$$W_2 = 0 \tag{5.32}$$

If the signal is an inverse-repeat MLB signal, then the shift-and-add property of (5.22) applies, so that the contribution of Y_3 can be determined as follows.

Contribution of Y_3

The first, second and third lines of (5.11) contribute to the terms

$$W_{3A}(iT) = \frac{Acb^2}{1-a}\left[a^{i-i_{1,1}}\left(1+\frac{1}{N}\right)-\left(\frac{(-1)^{i-i_{1,1}}}{N(1+a)}\right)\right] \qquad i = i_{1,1}, i_{1,1}+1, i_{1,1}+2,\ldots \quad (5.33)$$

$$W_{3B}(iT) = \frac{Acb^2}{1-a}\left[a^{1+i-i_{1,2}}\left(1+\frac{1}{N}\right)-\left(\frac{a(-1)^{i-i_{1,2}}}{N(1+a)}\right)\right] \qquad i = i_{1,2}, i_{1,2}+1, i_{1,2}+2,\ldots \quad (5.34)$$

$$W_{3C}(iT) = \frac{Acb^2}{1-a}\left[a^{1+i-i_{2,1}}\left(1+\frac{1}{N}\right)-\left(\frac{a(-1)^{i-i_{2,1}}}{N(1+a)}\right)\right] \qquad i = i_{2,1}, i_{2,1}+1, i_{2,1}+2,\ldots \quad (5.35)$$

respectively. Thus the total contribution of Y_3 is

$$W_3(iT) = \frac{Acb^2}{1-a}\left[\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a^{q+r-2+i-i_{q,r}}\left(1+\frac{1}{N}\right)-\left(\frac{(-1)^{i-i_{q,r}}}{N(1-a)^2(1+a)}\right)\right]$$

$$i = i_{q,r}, i_{q,r}+1, i_{q,r}+2,\ldots \quad (5.36)$$

The zero order term in the crosscorrelation function is a sum of the two terms

$$\Phi_{0A}(iT) = -\frac{Ac(-1)^i}{N(1-a)(1+a)} \qquad i = 0, 1, 2, \ldots \quad (5.37A)$$

$$\Phi_{0B}(iT) = -\frac{Acb^2(-1)^{i-i_{q,r}}}{N(1-a)^3(1+a)} \qquad i = i_{q,r}, i_{q,r}+1, i_{q,r}+2,\ldots \quad (5.37B)$$

The linear, quadratic and cubic terms in the crosscorrelation function are

$$\Phi_1(iT) = \frac{Ac}{1-a}a^i\left(1+\frac{1}{N}\right) \qquad i = 0, 1, 2, \ldots \quad (5.38)$$

$$\Phi_2 = 0 \quad (5.39)$$

$$\Phi_3(iT) = \frac{Acb^2}{1-a}\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a^{q+r-2+i-i_{q,r}}\left(1+\frac{1}{N}\right) \qquad i = i_{q,r}, i_{q,r}+1, i_{q,r}+2,\ldots \quad (5.40)$$

respectively.

5.4 Wiener Process with First Order Dynamics

5.4.1 Derivation of Crosscorrelation Function using Maximum Length Binary Signals

The Wiener model used for comparison with the direction-dependent process is shown in Figure 5.1. This model consists of a constant path, a linear path, a quadratic path and a cubic path. The quadratic path may be added to or subtracted from the total output depending on the dynamics of the direction-dependent model (and this will be discussed in Section 5.5.2). It is shown in [29] that each nonlinear path in the model contributes to terms that corrupt the crosscorrelation function obtained for the linear path alone, and the occurrences of discontinuities in the total crosscorrelation function resulting from these terms may be accounted for by (5.17) and (5.22).

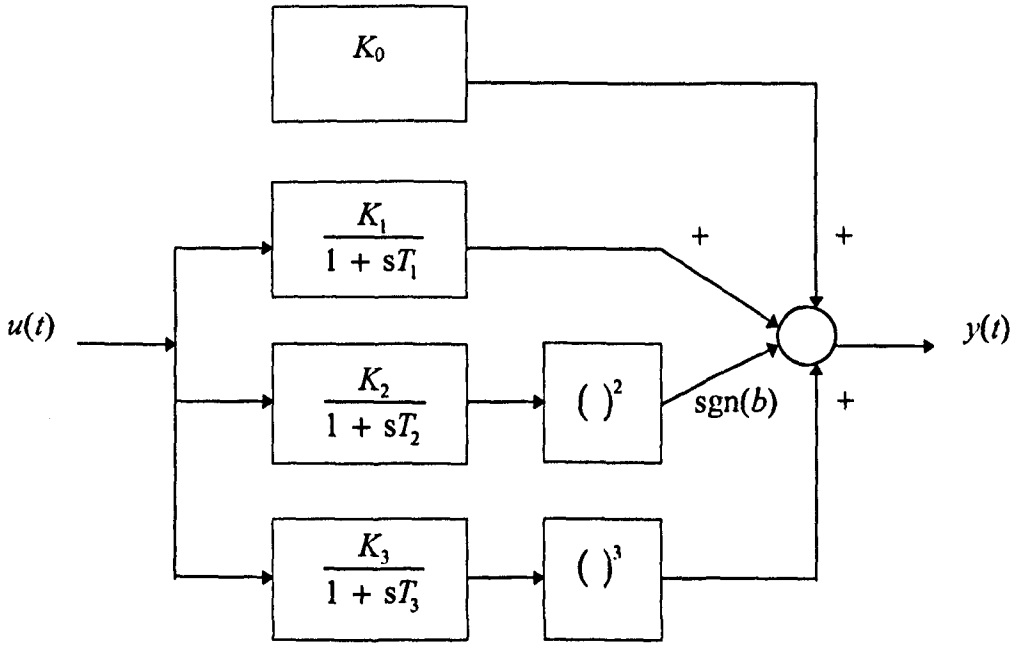

Figure 5.1. Structure of the Wiener model with first order dynamics.

Using an MLB signal, the contributions of the constant path, the linear path, the quadratic path and the cubic path to the crosscorrelation function are :

Contribution of constant path

$$Z_0 = \frac{K_0}{N} \tag{5.41}$$

since the average value of the input signal is $1/N$.

Contribution of linear path

$$Z_1(iT) = K_1(1-a_1)\left[a_1^i\left(1+\frac{1}{N}\right)-\left(\frac{1}{N(1-a_1)}\right)\right] \qquad i = 0, 1, 2, \ldots \tag{5.42}$$

where $a_1 = \exp(-T/T_1)$ (5.43)

The transfer function of the digital simulator for the linear path of the Wiener model is $K_1\frac{1-a_1}{z-a_1}$, where a_1 is the response decrement. This gives the impulse response as

$$1\text{-}a_1, (1\text{-}a_1)\,a_1, (1\text{-}a_1)\,a_1^2, \ldots\ldots$$

as there is no time delay between the input and the output in the simulation.

Contribution of quadratic path

The impulse response of the quadratic path is

$$1\text{-}a_2, (1\text{-}a_2)\,a_2, (1\text{-}a_2)\,a_2^2, \ldots\ldots$$

where $a_2 = \exp(-T/T_2)$ (5.44)

The output of the quadratic path is

$$\begin{aligned} V_2 = [\mathrm{sgn}(b)]K_2^2(1-a_2)^2(u_t^2 + a_2^2u_{t-1}^2 + a_2^4u_{t-2}^2 + a_2^6u_{t-3}^2 + \ldots \\ +2(a_2u_tu_{t-1} + a_2^3u_{t-1}u_{t-2} + a_2^5u_{t-2}u_{t-3} + \ldots \\ +a_2^2u_tu_{t-2} + a_2^4u_{t-1}u_{t-3} + \ldots \\ +a_2^3u_tu_{t-3} + \ldots \\ +\ldots)) \end{aligned} \tag{5.45}$$

The first line of (5.45) contributes to a constant term.

$$Z_{2A} = \frac{[\mathrm{sgn}(b)]K_2^2(1-a_2)^2}{N(1-a_2^2)} \tag{5.46}$$

The rest of the lines in (5.45) contribute to

$$Z_{2B}(iT) = -2[\mathrm{sgn}(b)]K_2{}^2(1-a_2)^2\left[\sum_{r=1}^{N-1} a_2{}^{r+2(i-i_r)}\left(1+\frac{1}{N}\right)-\left(\frac{1}{N(1-a_2{}^2)(1-a_2)}\right)\right] \tag{5.47}$$

where r and i_r are given in (5.17).

Contribution of cubic path

The impulse response of the cubic path is

$$1\text{-}\,a_3,\ (1\text{-}\,a_3)\,a_3,\ (1\text{-}\,a_3)\,a_3{}^2,\ \ldots\ldots$$

where $a_3 = \exp(-T/T_3)$ (5.48)

The output of the cubic path is

$$\begin{aligned} V_3 = K_3{}^3(1-a_3)^3(&u_t + a_3{}^3u_{t-1} + a_3{}^6u_{t-2} + a_3{}^9u_{t-3}+\ldots \\ &+3((a_3{}^2 + a_3{}^4 + a_3{}^6 + a_3{}^8\ldots)u_t \\ &+(a_3 + a_3{}^5 + a_3{}^7 + a_3{}^9\ldots)u_{t-1} \\ &+(a_3{}^2 + a_3{}^4 + a_3{}^8 + a_3{}^{10}\ldots)u_{t-2} \\ &+(a_3{}^3 + a_3{}^5 + a_3{}^7 + a_3{}^{11}\ldots)u_{t-3}+\ldots) \\ &+6(a_3{}^3u_tu_{t-1}u_{t-2} + a_3{}^6u_{t-1}u_{t-2}u_{t-3}+\ldots \\ &+a_3{}^4u_tu_{t-1}u_{t-3}+\ldots \\ &+a_3{}^5u_tu_{t-2}u_{t-3}+\ldots \\ &+\ldots)) \end{aligned} \tag{5.49}$$

since $u_t{}^2 = u_{t-1}{}^2 = u_{t-2}{}^2 = \ldots = 1$.

The first line of (5.49) contributes to a linear term

$$Z_{3A}(iT) = K_3{}^3(1-a_3)^3\left[a_3{}^{3i}\left(1+\frac{1}{N}\right)-\left(\frac{1}{N(1-a_3{}^3)}\right)\right] \quad i = 0, 1, 2, \ldots \tag{5.50}$$

The second to fifth lines of (5.49) contribute to

$$Z_{3B}(iT) = 3K_3^3(1-a_3)^3 a_3^i([1+a_3^2+a_3^4+a_3^6+\ldots]-a_3^{2i})$$

$$= 3K_3^3(1-a_3)^3 a_3^i\left(\frac{1}{1-a_3^2}-a_3^{2i}\right) \qquad i = 0, 1, 2, \ldots \qquad (5.51)$$

when considering only the on-peak value of the autocorrelation of the input signal. Taking into account the off-peak value,

$$Z_{3B}(iT) = 3K_3^3(1-a_3)^3\left[\frac{a_3^i}{(1-a_3^2)}\left(1+\frac{1}{N}\right)-\frac{1}{N(1-a_3)(1-a_3^2)}\right.$$

$$\left.-a_3^{3i}\left(1+\frac{1}{N}\right)+\frac{1}{N(1-a_3^3)}\right] \qquad i = 0, 1, 2, \ldots \qquad (5.52)$$

Line 6 and subsequent lines of (5.49) contribute to

$$Z_{3C}(iT) = 6K_3^3(1-a_3)^3\left[\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a_3^{2q+r+3(i-i_{q,r})}\left(1+\frac{1}{N}\right)-\left(\frac{a_3^3}{N(1-a_3^3)(1-a_3^2)(1-a_3)}\right)\right] \qquad (5.53)$$

where q, r and $i_{q,r}$ are given in (5.22).

Grouping all the constant terms

$$\Psi_0 = \frac{1}{N}\left(K_0 - K_1 + [\mathrm{sgn}(b)]K_2^2 - K_3^3\right) \qquad (5.54)$$

The linear, quadratic and cubic terms in the crosscorrelation function are

$$\Psi_1(iT) = \left(1+\frac{1}{N}\right)\left(K_1(1-a_1)a_1^i + 3K_3^3\frac{(1-a_3)^2}{(1+a_3)}a_3^i - 2K_3^3(1-a_3)^3 a_3^{3i}\right)$$

$$i = 0, 1, 2, \ldots \qquad (5.55)$$

$$\Psi_2(iT) = -2[\mathrm{sgn}(b)]K_2^2(1-a_2)^2\sum_{r=1}^{N-1} a_2^{r+2(i-i_r)}\left(1+\frac{1}{N}\right) \qquad i = i_r, i_r+1, i_r+2, \ldots \qquad (5.56)$$

$$\Psi_3(iT) = 6K_3^3(1-a_3)^3\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a_3^{2q+r+3(i-i_{q,r})}\left(1+\frac{1}{N}\right) \qquad i = i_{q,r}, i_{q,r}+1, i_{q,r}+2, \ldots \qquad (5.57)$$

respectively.

5.4.2 Derivation of Crosscorrelation Function using Inverse-repeat Maximum Length Binary Signals

If an inverse-repeat MLB signal is used, the contributions of the linear path and the cubic path to the crosscorrelation function are :

Contribution of linear path

$$Z_1(iT) = K_1(1-a_1)\left[a_1{}^i\left(1+\frac{1}{N}\right)-\left(\frac{(-1)^i}{N(1+a_1)}\right)\right] \qquad i = 0, 1, 2, \ldots \tag{5.58}$$

Contribution of cubic path

The first line of (5.49) contributes to a linear term

$$Z_{3A}(iT) = K_3{}^3(1-a_3)^3\left[a_3{}^{3i}\left(1+\frac{1}{N}\right)-\left(\frac{(-1)^i}{N(1+a_3{}^3)}\right)\right] \qquad i = 0, 1, 2, \ldots \tag{5.59}$$

The second to fifth lines of (5.49) contribute to

$$Z_{3B}(iT) = 3K_3{}^3(1-a_3)^3\left[\frac{a_3{}^i}{(1-a_3{}^2)}\left(1+\frac{1}{N}\right)-\frac{(-1)^i}{N(1+a_3)(1-a_3{}^2)}\right.$$
$$\left.-a_3{}^{3i}\left(1+\frac{1}{N}\right)+\frac{(-1)^i}{N(1+a_3{}^3)}\right] \qquad i = 0, 1, 2, \ldots \tag{5.60}$$

Line 6 and subsequent lines of (5.49) contribute to

$$Z_{3C}(iT) = 6K_3{}^3(1-a_3)^3\left[\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a_3{}^{2q+r+3(i-i_{q,r})}\left(1+\frac{1}{N}\right)-\left(\frac{(-1)^{i-i_{q,r}}}{N(1+a_3{}^3)(1-a_3)}\right)\right] \tag{5.61}$$

where q, r and $i_{q,r}$ are given in (5.22).

The zero order term in the crosscorrelation function is a sum of two terms

$$\Psi_{0A}(iT) = -\frac{K_1(1-a_1)(-1)^i}{N(1+a_1)} \qquad i = 0, 1, 2, \ldots \tag{5.62A}$$

$$\Psi_{0B}(iT) = -\frac{6K_3{}^3(1-a_3)^2(-1)^{i-i_{q,r}}}{N(1+a_3{}^3)} \qquad i = i_{q,r}, i_{q,r}+1, i_{q,r}+2, \ldots \tag{5.62B}$$

The linear, quadratic and cubic terms in the crosscorrelation function are

$$\Psi_1(iT) = \left(1+\frac{1}{N}\right)\left(K_1(1-a_1)a_1^{\,i} + 3K_3{}^3\frac{(1-a_3)^2}{(1+a_3)}a_3^{\,i} - 2K_3{}^3(1-a_3)^3 a_3{}^{3i}\right)$$

$$i = 0, 1, 2, \ldots \tag{5.63}$$

$$\Psi_2 = 0 \tag{5.64}$$

$$\Psi_3(iT) = 6K_3{}^3(1-a_3)^3\sum_{q=1}^{N-2}\sum_{r=1}^{N-1-q} a_3{}^{2q+r+3(i-i_{q,r})}\left(1+\frac{1}{N}\right)$$

$$i = i_{q,r}, i_{q,r}+1, i_{q,r}+2, \ldots \tag{5.65}$$

respectively.

5.5 Comparison between First Order Direction-dependent and Wiener Processes

5.5.1 Optimisation Procedure

The Wiener process can be made very similar to the direction-dependent process by matching the constant term, the linear term, the quadratic term and the cubic term separately. The optimisation procedure involved uses the function *fminunc* in the MATLAB Optimization Toolbox [61]. The default algorithm is based on least squares optimisation using Gauss-Newton method [63, 64]. (The function *fminsearch* [65] which is in MATLAB, but not in the above toolbox, can be used as an alternative. The algorithm is based on the Nelder-Mead simplex method [61, 66 - 69]. However, this function was not used in this chapter as the options available to the user are less than that in *fminunc*.)

Different methods were used to optimise the parameters of the Wiener model. This allowed comparison between the results obtained using different techniques and was used as a check for possible errors in the simulation. Six methods were used and these were minimising the

(a) sum of squares of the error in the crosscorrelation functions

(b) sum of the absolute error in the crosscorrelation functions

(c) sum of squares of the error in the outputs

(d) sum of the absolute error in the outputs
(e) sum of squares of the distance between the points in the frequency response, taking into account both magnitude and phase
(f) sum of the distance between the points in the frequency response, taking into account both magnitude and phase

Initial values were specified for each parameter before the start of the optimisation procedure and these were obtained through observation from initial simulations. The quality of the models obtained can be similarly evaluated using the quantities minimised in methods (a) to (f). These quantities will be denoted by the alphabets A to F, corresponding to the methods (in lowercase) in which they are minimised.

5.5.2 Comparison using Maximum Length Binary Signals

To illustrate the theory, a first order process was chosen in which $T_U = 3T$, $T_D = 12T$, $K_U = 4$ and $K_D = 1$ (Process A). For this process $A = 2.5$, $F = 1.5$, $a = 0.818$, $b = -0.102$ and $c = 0.0227$ from equations (5.3A), (5.3B), (3.4B), (3.4C) and (5.14) respectively.

Let the process be perturbed by an MLB signal $u(t)$ with period $127T$, generated from an MLB sequence $s(i)$ with characteristic polynomial $f(D) = 1 \oplus_2 D \oplus_2 D^4 \oplus_2 D^6 \oplus_2 D^7$, and the zero elements of $s(i)$ converted to -1 and the unity elements converted to +1. The signal has 64 clock-pulse intervals at signal level +1 and 63 clock-pulse intervals at signal level -1. Second order terms will cause discontinuities in the crosscorrelation function as given in (5.26). The starting positions of the most significant of these terms are given in Table 5.1. There are several MLB signals with period $127T$, and this particular one was selected because the starting positions listed in Table 5.1 are the furthest removed from the origin compared to other MLB signals with this period [29]; this makes the additional peaks due to second order terms easier to detect by eye since they are less likely to occur on the main (linear dynamics) peak. Third order terms also contribute to discontinuities in the crosscorrelation function as given in (5.27). However, the amplitudes of these discontinuities are smaller compared with those

caused by second order terms. The starting position of the most significant of these terms are given in Table 5.2. Similarly, higher order terms will cause such discontinuities but their amplitudes are generally small enough to be neglected.

r	1	2	3	4	5	6	7	8	9	10
i_r	89	51	21	102	28	42	33	77	62	56
r	11	12	13	14	15	16	17	18	19	20
i_r	111	84	60	66	35	27	103	124	83	112

Table 5.1. Start positions of discontinuities in the crosscorrelation function due to second order terms.

q	1	1	2	1	2	3
r	1	2	1	3	2	1
$i_{q,r}$	59	85	113	107	118	95

Table 5.2. Start positions of discontinuities in the crosscorrelation function due to third order terms.

The input-output crosscorrelation function of the direction-dependent process is shown in Figure 5.2 (plotted as circles). For this process and this signal, from equation (5.24), the constant term $\Phi_0 = -0.00787$ while from equation (5.25), the linear dynamics term is given by

$$\Phi_1(iT) = 0.314(0.818)^i \qquad i = 0, 1, 2, \ldots$$

From equation (5.26) and Table 5.1, the second order dynamics term $\Phi_2(iT)$ is given by

$$\begin{aligned}
&0.0320(0.818)^{i-89}, && i = 89, 90, 91, \ldots \\
&+ 0.0320(0.818)(0.818)^{i-51}, && i = 51, 52, 53, \ldots \\
&+ 0.0320(0.818)^2(0.818)^{i-21}, && i = 21, 22, 23, \ldots \\
&+ 0.0320(0.818)^3(0.818)^{i-102}, && i = 102, 103, 104, \ldots \\
&+ \ldots
\end{aligned}$$

From equation (5.27) and Table 5.2, the third order dynamics term $\Phi_3(iT)$ is given by

$$
\begin{aligned}
&0.00325(0.818)^{i-59}, && i = 59, 60, 61, \ldots \\
&+ 0.00325(0.818)(0.818)^{i-85}, && i = 85, 86, 87, \ldots \\
&+ 0.00325(0.818)(0.818)^{i-113}, && i = 113, 114, 115, \ldots \\
&+ 0.00325(0.818)^2(0.818)^{i-107}, && i = 107, 108, 109, \ldots \\
&+ 0.00325(0.818)^2(0.818)^{i-118}, && i = 118, 119, 120, \ldots \\
&+ 0.00325(0.818)^2(0.818)^{i-95}, && i = 95, 96, 97, \ldots \\
&+ \ldots
\end{aligned}
$$

These pattern can be confirmed by inspection of Figure 5.2. Note that the amplitude of the largest second order peak (at lag 89) is approximately 10% of the amplitude of the linear peak (at lag zero), while the amplitude of the largest third order peak (at lag 59) is approximately 1% of the amplitude of the linear peak.

Using optimisation, the parameters for the corresponding Wiener model are given in Table 5.3 and the error measurements are given in Table 5.4. From Table 5.3, it can be seen that the results obtained using different methods are in reasonably close agreement with one another. Since the dynamics in the positive direction are faster than those in the negative direction, b is negative. The sign in the quadratic path needs to be negative in order that the peaks due to the second order terms will be positive. The crosscorrelation function of the Wiener process is shown as plusses in Figure 5.2. The outputs of the these processes are shown in Figure 5.3.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(a)	2.97	1.63	0.98	0.80	4.86	6.90	8.90
(b)	2.96	1.62	0.95	0.78	4.81	6.51	8.30
(c)	2.97	1.60	0.94	0.82	4.78	6.44	7.96
(d)	2.98	1.60	0.96	0.87	4.82	6.36	8.11
(e)	2.97	1.60	0.94	0.82	4.78	6.43	7.88
(f)	2.97	1.59	0.87	0.70	4.75	6.14	6.79

Table 5.3. Parameters of the Wiener model for Process A.

Method	A ($*10^{-4}$)	B	C ($*10^{-2}$)	D	E	F
(a)	3.01	0.154	3.86	1.77	4.90	18.5
(b)	2.80	0.145	5.31	2.20	6.74	18.9
(c)	2.69	0.146	3.41	1.67	4.33	16.6
(d)	2.90	0.154	4.10	1.67	5.21	19.4
(e)	2.70	0.146	3.41	1.67	4.34	16.7
(f)	3.07	0.160	4.08	1.71	5.19	15.7

Table 5.4. Error measurements of the Wiener model for Process A.

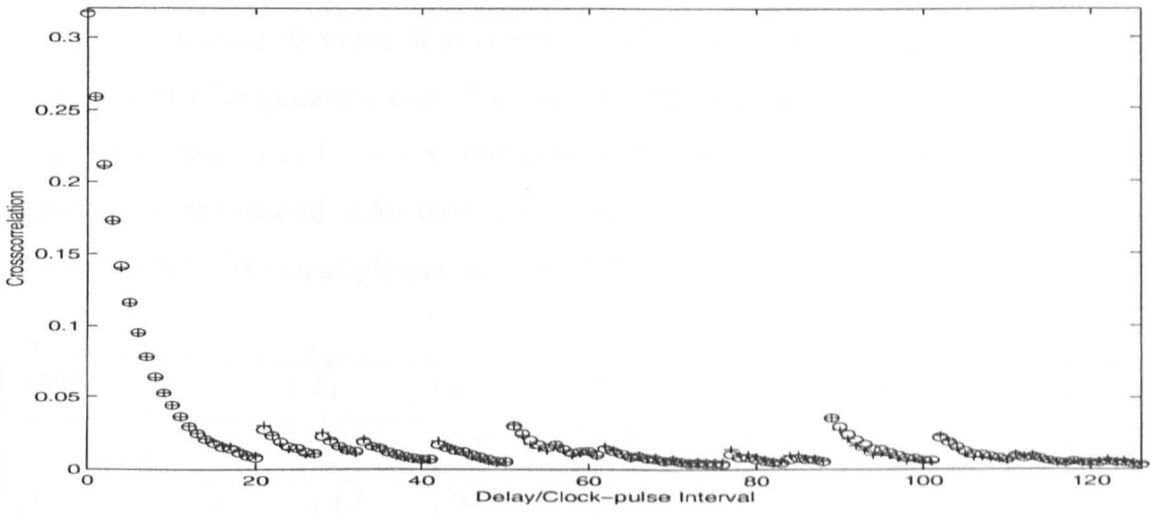

Figure 5.2. Crosscorrelation functions of Process A (circles) and its corresponding Wiener model (plusses).

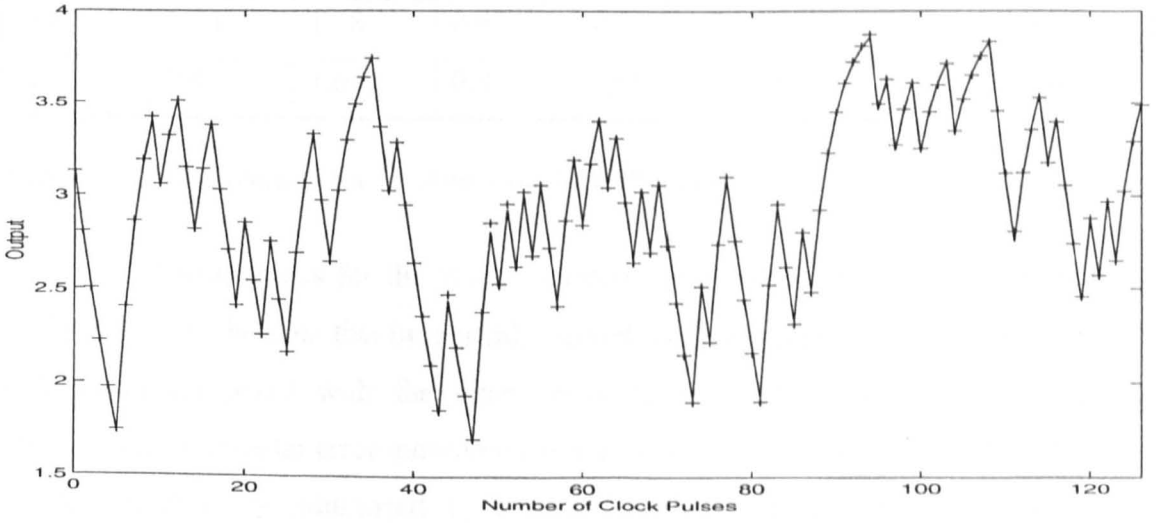

Figure 5.3. Outputs of Process A (solid line) and its corresponding Wiener model (plusses).

If the dynamics of the direction-dependent process are faster in the negative direction instead, then b = 0.102, and the second order and higher even order terms will cause negative peaks in the crosscorrelation function. The constant, linear and third order terms remain unchanged. This is shown as circles in Figure 5.4 using the same input signal but with the process having $T_U = 12T$, $T_D = 3T$, $K_U = 4$ and $K_D = 1$ (Process B).

For the corresponding Wiener model, the parameters are given in Table 5.5. Again, close agreement is observed in the results obtained using different methods, except in the values of K_3 and T_3. This is due to the fact that the contribution of the cubic terms in the crosscorrelation function of the direction-dependent process is small and hence, the fitting of the cubic path in the Wiener process is less accurate than the constant path, the linear path and the quadratic path. This can also be seen from the larger variation in the values of K_3 and T_3 in Table 5.3. The sign in the quadratic path is positive so that the peaks due to the second order terms are negative. The crosscorrelation function of the Wiener model is shown as plusses in Figure 5.4.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(a)	0.03	1.67	0.97	0.78	4.96	6.90	8.90
(b)	0.04	1.67	0.96	0.63	4.95	6.62	6.27
(c)	0.03	1.68	0.96	0.60	4.96	6.55	7.00
(d)	0.03	1.69	0.95	0.55	4.94	6.40	7.04
(e)	0.03	1.68	0.97	0.51	4.98	6.65	4.99
(f)	0.04	1.69	0.89	0.59	5.00	6.24	6.02

Table 5.5. Parameters of the Wiener model for Process B.

The error measurements for the Wiener models are shown in Table 5.6. From this and Table 5.4, it can be seen that in general, methods (c) and (e) give models with relatively low errors compared with the other methods. It is also interesting to note that minimising a particular error measurement does not necessarily result in a Wiener model where the quantity minimised is actually smaller than that obtained using other techniques. To complicate matters, the errors obtained depend on the direction-

dependent process parameters and the initial values entered during the optimisation. Due to the large number of parameters to be optimised, there are likely to be many local minima in the objective functions minimised.

Method	A (*10^{-4})	B	C (*10^{-2})	D	E	F
(a)	3.37	0.158	4.33	1.83	5.50	18.4
(b)	2.93	0.144	4.57	2.04	5.80	18.6
(c)	2.93	0.145	3.71	1.71	4.72	17.8
(d)	3.00	0.153	3.80	1.68	4.82	18.2
(e)	2.88	0.147	3.65	1.71	4.64	17.6
(f)	3.33	0.160	4.32	1.93	5.48	16.9

Table 5.6. Error measurements of the Wiener model for Process B.

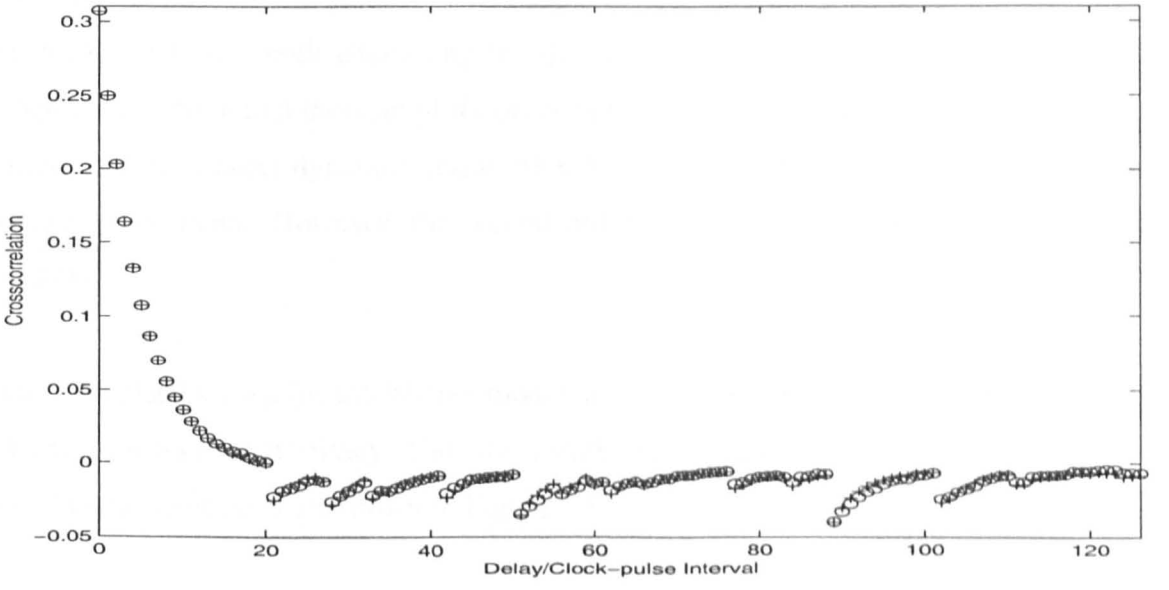

Figure 5.4. Crosscorrelation functions of Process B (circles) and its corresponding Wiener model (plusses).

5.5.3 Comparison using Inverse-repeat Maximum Length Binary Signals

Inverse-repeat signals contain only odd harmonics thus allowing the effects of odd and even order nonlinearities to be separated in the system output and identified separately.

This can be done by considering the fact that the former (including the linear dynamics) and the latter contribute to an inverse-repeat component and a repeat-repeat component respectively. Hence contributions from the odd order terms (for the first half of the period) could be obtained by subtracting the second half of the signal from the first and then dividing the result by two. Similarly, contributions from the even order terms could be obtained by adding the two halves of the signal and dividing the result by two. In the frequency domain, the equivalent procedure is carried out by setting either the odd or the even harmonics to zero. The use of such signals will result in an increase in the consistency and accuracy of the measurements [70]. However, there is also an associated disadvantage which is that it is not possible to separate the odd and even order terms in the crosscorrelation function. Consequently, only methods (c) to (f) can be used and the error measurement only needs to be calculated for the corresponding quantities (*C* to *F*).

For Process A, from equations (5.37A) and (5.37B), the zero order term consists of the sum of two terms, both alternating in sign and of amplitudes 0.00135 and 0.000423 respectively. Note that these amplitudes are both very small. From (5.38) and (5.40), the linear and third order dynamics terms are the same as that when using the non inverse-repeat MLB signal. However, the second order dynamics term is zero (from equation (5.39)).

The optimised values for the Wiener model are shown in Tables 5.7 and 5.8 for Process A and Process B respectively. The crosscorrelation functions of the direction-dependent and Wiener processes are shown in Figure 5.5.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(c)	2.97	1.64	1.01	0.81	4.88	7.22	8.90
(d)	2.98	1.62	1.05	0.96	4.83	7.82	11.97
(e)	2.97	1.64	1.01	0.84	4.87	7.22	9.60
(f)	2.96	1.64	0.89	0.77	4.88	6.36	8.58

Table 5.7. Parameters of the Wiener model for Process A using an inverse-repeat signal.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(c)	0.03	1.64	1.01	0.81	4.88	7.22	8.90
(d)	0.02	1.62	1.05	0.96	4.83	7.82	11.94
(e)	0.03	1.64	1.01	0.84	4.87	7.22	9.60
(f)	0.04	1.64	0.89	0.77	4.88	6.36	8.58

Table 5.8. Parameters of the Wiener model for Process B using an inverse-repeat signal.

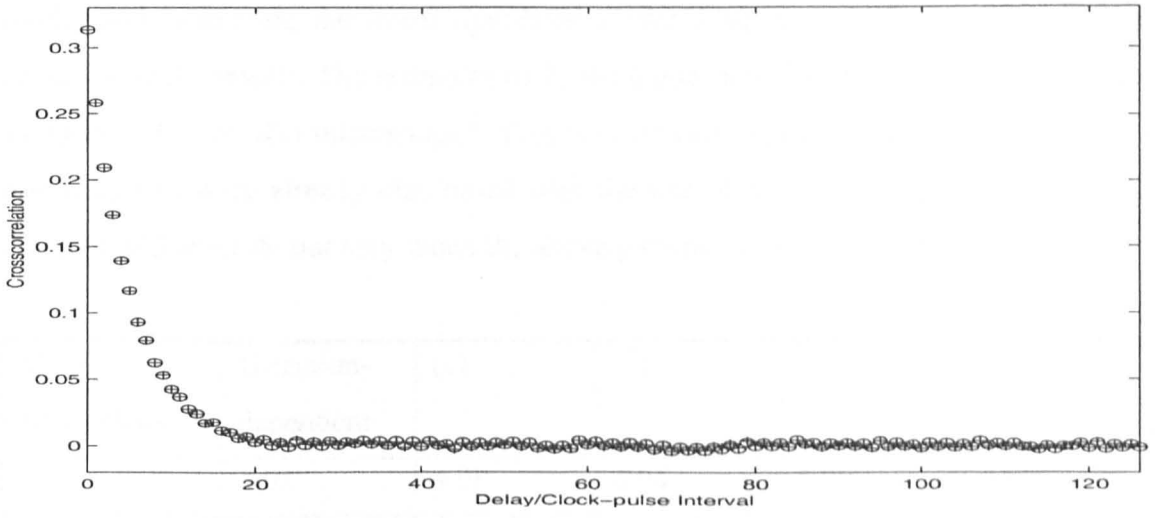

Figure 5.5. Crosscorrelation functions of Process A (or B) (circles) and its corresponding Wiener model (plusses) using an inverse-repeat signal.

The error measurements for the Wiener models in Table 5.7 are given in Table 5.9. Very similar results were obtained for those in Table 5.8. It may seem surprising that the values in Table 5.9 are larger than the corresponding values in Table 5.4. However, these should not be compared because the signals used are different in length and harmonic content.

Method	C	D	E	F
(c)	0.126	4.52	32.1	52.8
(d)	0.136	4.48	34.6	58.3
(e)	0.126	4.51	32.1	53.0
(f)	0.182	5.48	46.2	48.7

Table 5.9. Error measurements of the Wiener model for Process A using an inverse-repeat signal.

5.5.4 Estimation of Linear Dynamics

The linear dynamics for both direction-dependent and Wiener processes were estimated using the MATLAB algorithms ARX, ARMAX and ELiS, and correlation analysis with least squares. (These were discussed in Sections 4.2.1, 4.2.2 and 4.2.3.) Results obtained are given in Table 5.10. These are either equal or close to the theoretical result. An inverse-repeat MLB signal was used both for optimising the parameters of the Wiener model and estimating the linear dynamics as this leads to a higher consistency and accuracy of the results. The estimates of T_C were unchanged when obtained with T_U and T_D (and/or K_U and K_D) interchanged. This is to be expected as the effects of even order nonlinearities were already eliminated with the use of an inverse-repeat signal, while those of odd order do not vary when the above parameter values are interchanged.

Models/ Algorithms	Direction-dependent	(c)	(d)	(e)	(f)
ARX	4.98	4.98	4.94	5.00	5.00
ARMAX	4.98	4.98	4.96	5.01	5.02
ELiS	5.00	5.00	5.00	5.00	4.98
Correlation analysis	4.99	4.98	4.95	4.97	4.96
Theoretical	4.99				

Table 5.10. Estimated combined time constant T_C/T for Process A (or B) and its corresponding Wiener model optimised using methods (c) to (f).

5.6 Extension to Second Order Processes

For a second order direction-dependent process, the peaks in the crosscorrelation function do not necessarily have the same sign as in the first order case. Additionally, some of these may be undetectable due to their small amplitude (see Section 3.4.2.1). Thus, it is difficult to obtain a good match in the crosscorrelation function using a Wiener model with second order dynamics (as shown in Figure 5.6) since all the

discontinuities in the crosscorrelation function caused by a particular path are in the same direction. The sign of the quadratic path could not be determined theoretically and the one which gave a better fit was used (for each individual case).

The optimisation procedure is similar to that described in Section 5.5.1. The second order dynamics paths in the Wiener model were fixed at critical damping since this increased the reliability of the results obtained due to there being less parameters to optimise. This was also followed by an improvement in the speed of the simulation runs. Furthermore, the results often did not converge when these paths were fixed otherwise.

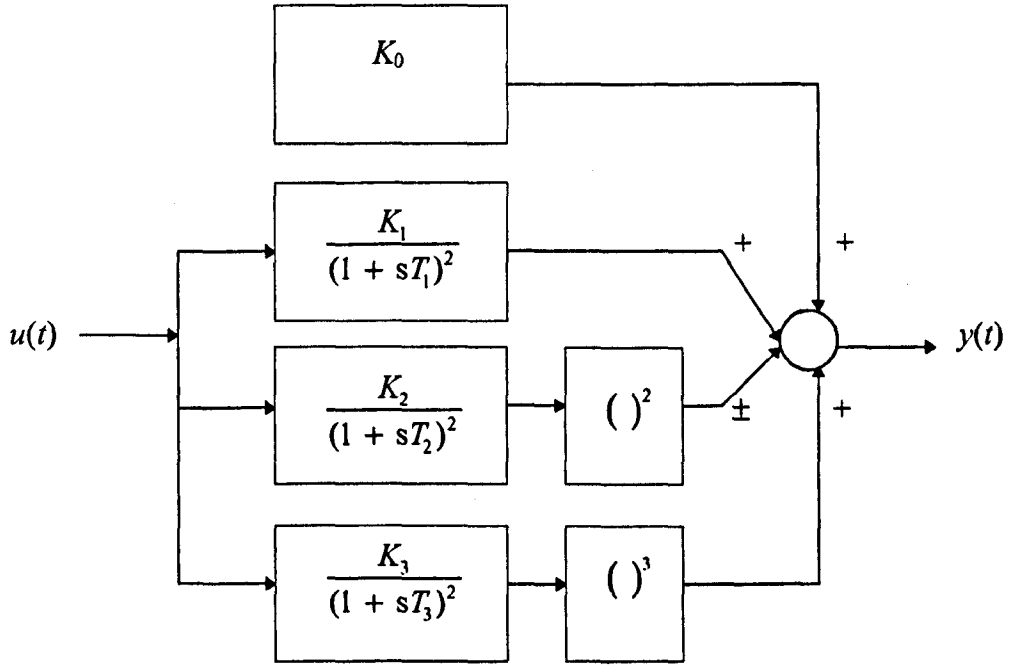

Figure 5.6. Structure of the Wiener model with second order dynamics.

An example is illustrated for a second order process with $\zeta = 1$ in both directions, $\omega_{nU} = 1/T$ ($T_U = T$, T) and $\omega_{nD} = 0.25/T$ ($T_D = 4T$, $4T$) (Process C). The gain is unity in both directions. Results obtained using an MLB signal are given in Table 5.11 and the sign of the quadratic path is positive. The error measurements are given in Table 5.12. These are in general larger than those given in Table 5.4, since the matching was less good compared with the first order case.

The crosscorrelation functions of the direction-dependent and the Wiener processes are illustrated in Figure 5.7. From the figure, is can be seen that the Wiener model matches the main (linear) peak of the direction-dependent process reasonably well. However, the rest of the peaks do not give such a good match. Also, the crosscorrelation function of the Wiener model is relatively flat compared to that of the direction-dependent process. The outputs of both processes are shown in Figure 5.8.

Results obtained using the corresponding inverse-repeat signal are given in Table 5.13. Again, these are seen to be more consistent then those obtained using the original (non inverse-repeat) signal. The corresponding values of the quantities C to F are shown in Table 5.14.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(a)	0.18	1.00	0.22	-0.23	2.09	0.82	1.48
(b)	0.27	1.04	0.14	-0.37	2.15	0.40	1.78
(c)	0.17	1.02	0.47	-0.68	2.06	0.83	1.54
(d)	0.16	0.99	0.26	-0.28	2.00	0.84	1.39
(e)	0.17	1.02	0.22	-0.31	2.06	0.83	1.53
(f)	0.18	1.06	0.18	-0.27	2.11	0.82	1.45

Table 5.11. Parameters of the Wiener model for Process C.

Method	A (*10^{-2})	B	C	D	E	F
(a)	1.36	0.95	1.89	12.1	140	119
(b)	1.46	0.96	2.16	13.9	274	124
(c)	0.96	0.84	1.21	10.3	154	104
(d)	1.36	0.96	2.06	12.5	262	121
(e)	1.41	0.96	2.06	12.6	261	122
(f)	1.53	1.01	2.14	12.9	272	125

Table 5.12. Error measurements of the Wiener model for Process C.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(c)	0.21	1.01	0.34	-0.13	2.19	0.71	1.92
(d)	0.19	1.01	0.38	-0.12	2.16	0.77	1.70
(e)	0.21	1.01	0.34	-0.13	2.19	0.72	1.91
(f)	0.21	0.97	0.34	-0.05	2.19	0.78	1.52

Table 5.13. Parameters of the Wiener model for Process C using an inverse-repeat signal.

Method	C	D	E	F
(c)	2.56	20.8	650	268
(d)	2.65	20.6	672	275
(e)	2.56	20.7	650	268
(f)	2.52	20.5	639	262

Table 5.14. Error measurements of the Wiener model for Process C using an inverse-repeat signal.

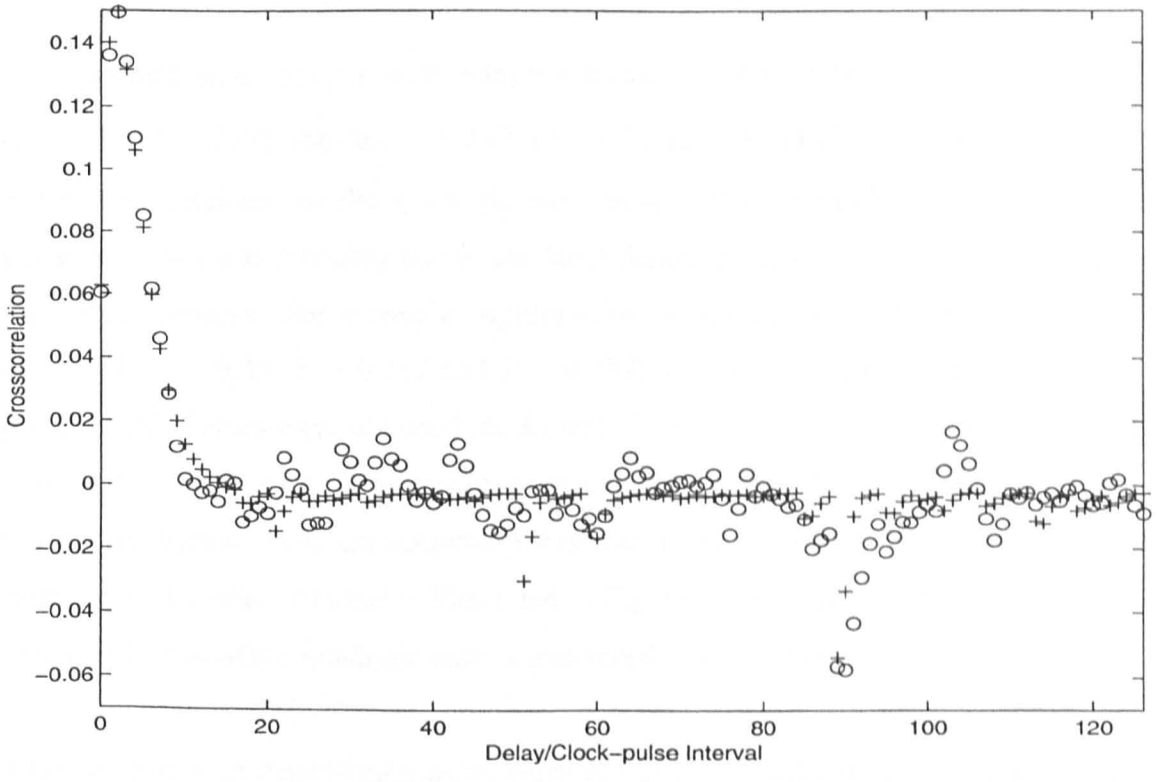

Figure 5.7. Crosscorrelation functions of Process C (circles) and its corresponding Wiener model (plusses).

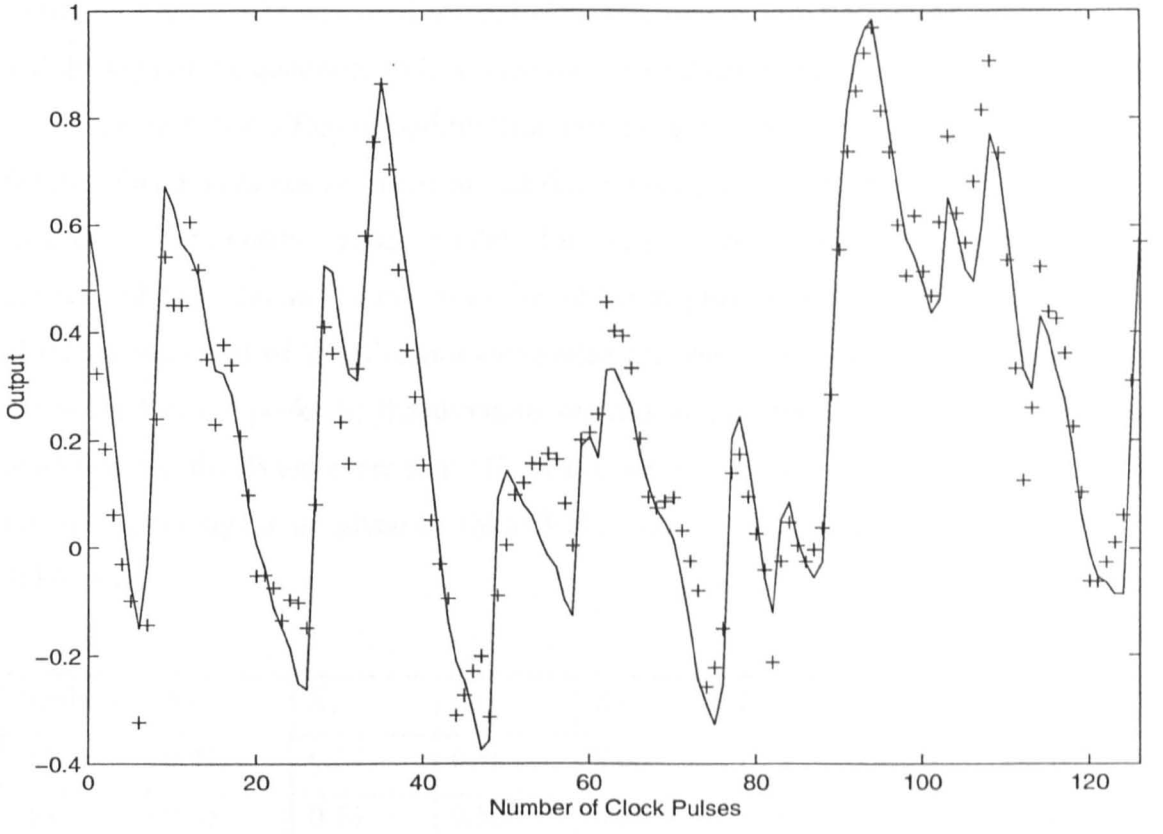

Figure 5.8. Outputs of Process C (solid line) and its corresponding Wiener model (plusses).

For a second order process with damping factor $\zeta = 2$ in both directions, $\omega_{nU} = 5/T$ ($T_U = 0.05T$, $0.75T$) and $\omega_{nD} = 1.25/T$ ($T_D = 0.21T$, $2.99T$) (Process D), a very poor match was obtained for the quadratic path using a Wiener model with second order dynamics. This was probably due to the large damping factor and the fast dynamics of the actual process. For example, optimisation using method (c) gave $K_0 = 0.13$, $K_1 = 0.71$, $K_2 = 0.48$, $T_1 = 0.51T$ and $T_2 = 0.03T$, and the sign of the quadratic path was positive. (No values were obtained for K_3 and T_3 because the simulation was conducted without the cubic path as a very poor match was obtained for the quadratic path.) Similar problems were encountered using the other methods. The crosscorrelation function of the Wiener model is illustrated in Figure 5.9 as a solid line. From the figure, the contribution of the quadratic path is undetectable due to the very small value of T_2.

Since the direction-dependent process behaves like a first order process (due to the large damping factor and the fast dynamics), a Wiener model with first order dynamics (as

shown in Figure 5.1) was used instead. The parameters found are given in Table 5.15 and the sign of the quadratic path is negative. The values of the errors are given in Table 5.16. The fact that different optimisation criteria gave approximately the same results for the Wiener parameters indicated that the Wiener process with first order dynamics is indeed a reasonably good model for this direction-dependent process. The crosscorrelation function is shown in Figure 5.9 as plusses, and a much closer match is observed with that of the direction-dependent process, shown as circles. In particular, the second order peaks in the direction-dependent process are now reasonably well modelled by the Wiener process. The parameters obtained using the corresponding inverse-repeat signal are given in Table 5.17, with the error measurements tabulated in Table 5.18.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(a)	0.41	0.57	0.56	0.65	1.29	2.30	1.10
(b)	0.35	0.55	0.52	0.68	1.26	2.17	1.15
(c)	0.43	0.57	0.57	0.65	1.29	2.31	1.07
(d)	0.42	0.54	0.52	0.65	1.28	2.22	0.97
(e)	0.43	0.57	0.57	0.65	1.29	2.32	1.10
(f)	0.42	0.58	0.57	0.64	1.30	2.37	1.08

Table 5.15. Parameters of the Wiener model for Process D.

Method	A (*10^{-4})	B	C (*10^{-1})	D	E	F
(a)	6.07	0.169	10.3	2.86	13.1	33.0
(b)	7.46	0.156	6.90	8.81	87.6	42.0
(c)	6.03	0.172	7.74	2.41	9.83	31.3
(d)	7.75	0.207	9.83	2.28	12.5	33.4
(e)	6.05	0.173	7.80	2.43	9.90	31.7
(f)	6.06	0.171	7.94	2.36	10.1	31.4

Table 5.16. Error measurements of the Wiener model for Process D.

Method	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
(c)	0.43	0.58	0.59	0.66	1.29	2.42	1.19
(d)	0.42	0.55	0.60	0.65	1.30	2.54	0.97
(e)	0.43	0.58	0.59	0.66	1.29	2.42	1.19
(f)	0.42	0.60	0.59	0.64	1.32	2.48	1.14

Table 5.17. Parameters of the Wiener model for Process D using an inverse-repeat signal.

Method	C	D	E	F
(c)	0.193	5.34	49.0	97.9
(d)	0.237	5.43	60.1	103.4
(e)	0.193	5.34	49.0	97.9
(f)	0.200	5.33	50.9	97.3

Table 5.18. Error measurements of the Wiener model for Process D using an inverse-repeat signal.

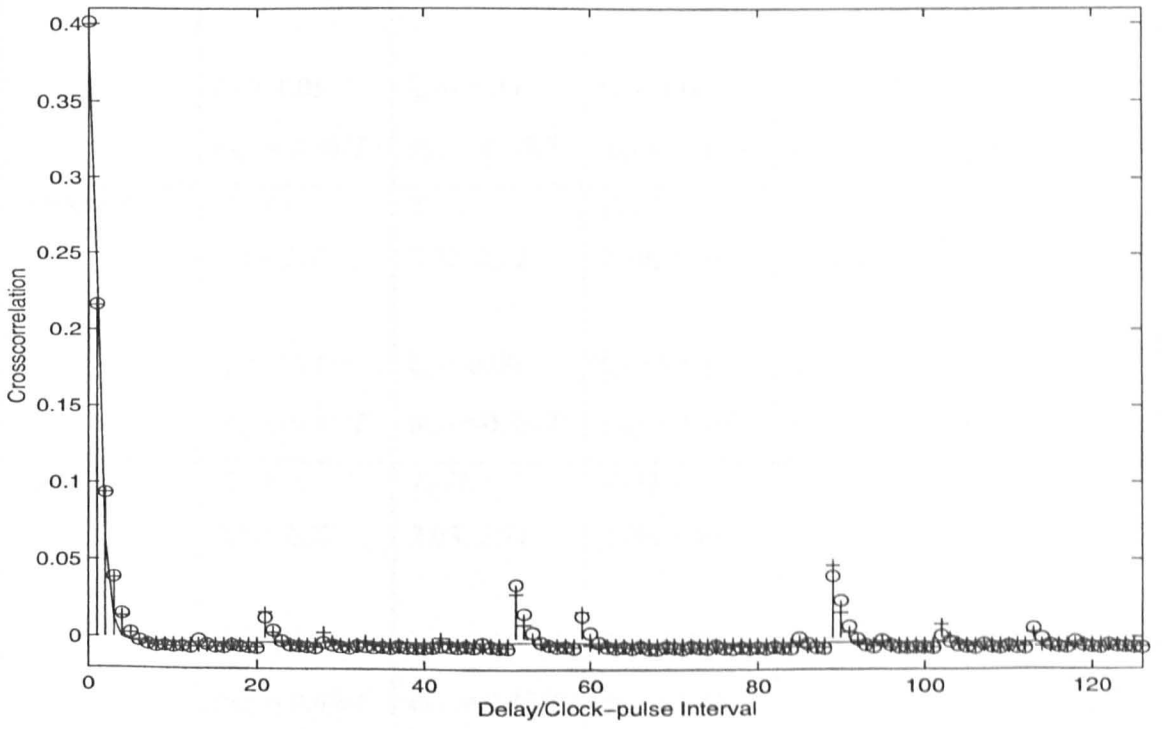

Figure 5.9. Crosscorrelation functions of Process D (circles), its corresponding Wiener model with second order dynamics (solid line) and that with first order dynamics (plusses).

The estimated linear dynamics for Processes C and D, and their corresponding Wiener models are given in Tables 5.19 and 5.20 respectively. These were obtained using an inverse-repeat signal. Correlation analysis with least squares was not used as this method does not give good results for second order processes (see Section 4.3.2). From Table 5.19, little variation is observed in the estimates either for the Wiener models obtained using different methods or using different estimation algorithms. This shows that all these optimisation methods as well as estimation algorithms can be effectively used to model and identify the direction-dependent process. In Table 5.20, the estimation algorithms were used to fit second order models to the direction-dependent process. However, the large damping factor made this computationally difficult and ELiS gave a result with a single time constant. (The Wiener models were fitted with first order models since they were genuinely first order processes.)

Models/ Algorithms	Direction-dependent	(c)	(d)	(e)	(f)
ARX	$T_C/T =$ 1.53, 2.80 $\zeta_C = 1.05$ $\omega_{nC} = 0.48/T$	$T_C/T =$ 2.22, 2.22 $\zeta_C = 1.00$ $\omega_{nC} = 0.45/T$	$T_C/T =$ 2.02, 2.37 $\zeta_C = 1.00$ $\omega_{nC} = 0.46/T$	$T_C/T =$ 2.22, 2.22 $\zeta_C = 1.00$ $\omega_{nC} = 0.45/T$	$T_C/T =$ 2.22, 2.22 $\zeta_C = 1.00$ $\omega_{nC} = 0.45/T$
ARMAX	$T_C/T =$ 1.39, 3.59 $\zeta_C = 1.11$ $\omega_{nC} = 0.45/T$	$T_C/T =$ 2.22, 2.22 $\zeta_C = 0.99$ $\omega_{nC} = 0.46/T$	$T_C/T =$ 2.19, 2.19 $\zeta_C = 0.99$ $\omega_{nC} = 0.46/T$	$T_C/T =$ 2.22, 2.22 $\zeta_C = 0.99$ $\omega_{nC} = 0.46/T$	$T_C/T =$ 2.22, 2.22 $\zeta_C = 0.99$ $\omega_{nC} = 0.46/T$
ELiS	$T_C/T =$ 2.37, 2.37 $\zeta_C = 0.96$ $\omega_{nC} = 0.44/T$	$T_C/T =$ 2.03, 2.41 $\zeta_C = 1.00$ $\omega_{nC} = 0.45/T$	$T_C/T =$ 2.00, 2.39 $\zeta_C = 1.00$ $\omega_{nC} = 0.46/T$	$T_C/T =$ 2.03, 2.41 $\zeta_C = 1.00$ $\omega_{nC} = 0.45/T$	$T_C/T =$ 2.03, 2.41 $\zeta_C = 1.00$ $\omega_{nC} = 0.45/T$

Table 5.19. Estimated combined dynamics for Process C and its corresponding Wiener model optimised using methods (c) to (f).

Models/ Algorithms	Direction-dependent	(c)	(d)	(e)	(f)
ARX	$T_C/T = 0.53, 1.15$ $\zeta_C = 1.08$ $\omega_{nC} = 1.28/T$	1.53	1.51	1.53	1.54
ARMAX	$T_C/T = 0.53, 1.15$ $\zeta_C = 1.08$ $\omega_{nC} = 1.29/T$	1.57	1.53	1.57	1.57
ELiS	$T_C/T = 1.47^*$	1.47	1.43	1.47	1.47

Table 5.20. Estimated combined dynamics for Process D and its corresponding first order Wiener model optimised using methods (c) to (f). *A second order model could not be estimated using ELiS in this case.

5.7 Conclusions

The identification of systems with direction-dependent dynamics was extended to systems with direction-dependent gains. The terms in the crosscorrelation function were developed for a first order process perturbed with an MLB signal and an inverse-repeat MLB signal as coherent patterns are not observed when other classes of binary pseudo-random signals are used.

For a first order direction-dependent process, a very good match can be obtained using a Wiener model with first order dynamics consisting of a constant path, a linear path, a quadratic path and a cubic path. The model parameters were chosen based on different criteria using the Optimization Toolbox. Results obtained through simulation were compared, and these also served to validate the theoretical analysis.

The linear dynamics for both direction-dependent and Wiener processes were estimated using the ARX, ARMAX, ELiS and correlation analysis algorithms. An inverse-repeat MLB signal was used because it eliminates the effects of even order nonlinearities, leading to a higher accuracy and consistency in the estimates obtained.

For a second order direction-dependent process, the extra peaks in the crosscorrelation function do not necessarily have the same sign as in the first order case. Thus, a Wiener model with second order dynamics may not provide a good match for the quadratic and cubic paths since all the peaks caused by the same path in the Wiener process will be in the same direction. However, if the damping factor of the actual process is large and the process has fast dynamics, a Wiener model with first order dynamics may be used instead, to provide a reasonable match to the direction-dependent process.

Chapter 6

Modelling of Direction-dependent Dynamic Processes : A Comparison of Wiener Models and Neural Networks

6.1 Abstract

The modelling of direction-dependent processes using the Wiener model has been considered in the previous chapter. An alternative model using a neural network architecture is also suggested in the literature. The objective of this chapter is to compare the two approaches for several different processes and for four different types of input signal - a pseudo-random binary signal, an inverse-repeat pseudo-random binary signal, a five-level maximum length signal generated from Galois field GF(5) and a multisine (sum of harmonics) signal. Experimental results on an electronic nose system will also be presented. For this process, the direction-dependent characteristic is observed between the input which is the strength of the chemical odour, and the output which is the voltage across the metal oxide semiconductor (MOS) sensor.

6.2 Neural Network Architecture

The neural network uses a 'semirecurrent' architecture shown in Figure 6.1 which was proposed in [40], and is based on, and modified from, the fully recurrent network [71]. In the figure, u is the input and y is the output of the network. The main advantage of this architecture is that the gradient of the process output is taken into account through feedback of the previous values of the predicted output thus enabling the direction-dependent characteristics to be captured.

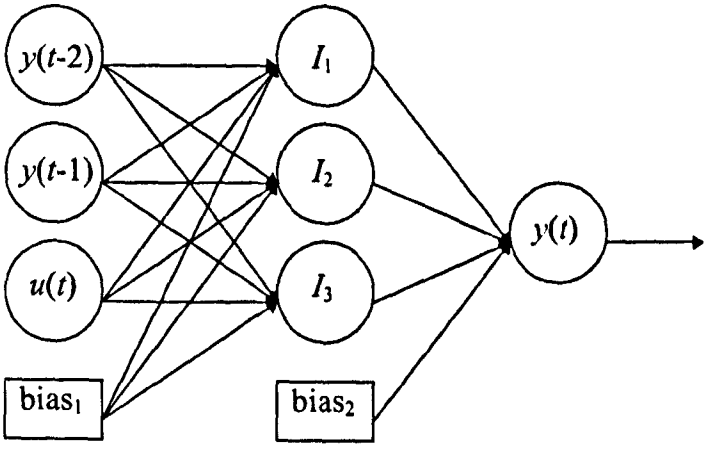

Figure 6.1. Architecture of the 'semirecurrent' neural network.

The neural network consists of one hidden layer with three neurons. The transfer function from the input to the hidden layer is a hyperbolic tangent sigmoid function given by

$$\text{tansig}(n) = \frac{2}{1 + \exp(-2n)} - 1 \tag{6.1}$$

The reason for using this transfer function is that it is a good trade-off for neural networks, where speed is important but not the exact shape of the transfer function. The hidden layer is linearly related to the output of the network.

In the implementation used in the present chapter, backpropagation [72 - 74] was used to update the parameters of the neural network. This computes the gradient in the weight space of an error measurement based on the mismatch between the actual performance and the desired performance. The training function was based on the Levenberg-Marquardt algorithm which has the advantage of being very fast. The performance measure used was the mean squared error between the actual output and the desired output. The neural network was implemented with the function *newff* using the Neural Network Toolbox [75] in MATLAB.

6.3 Modelling First Order Direction-dependent Process

In this section, twelve cases, involving three different processes and four different input signals will be considered; these are tabulated in Table 6.1. (Processes A and B were already defined in Chapter 5.) The Wiener system (shown in Figure 5.1) and the neural network (shown in Figure 6.1) will be used to model these processes and the quality of the fit will be compared.

In Cases 1 to 3, the processes were perturbed with an MLB signal with feedback from stages 1, 4, 6 and 7 (period $N = 127$) as was done in Section 5.5.2. The Wiener parameters are given in Table 6.2. (These were obtained in Section 5.5.2 by minimising the mean of squares of the error in the outputs. The sign of the quadratic path is negative for Processes A and E, and positive for Process B following the result in Section 5.4.1).

Case	Process	Input signal
1	Process E: $T_U = 3T$, $T_D = 12T$, $K_U = 1$, $K_D = 1$	Binary
2	Process B: $T_U = 12T$, $T_D = 3T$, $K_U = 4$, $K_D = 1$	Binary
3	Process A: $T_U = 3T$, $T_D = 12T$, $K_U = 4$, $K_D = 1$	Binary
4	Process E: $T_U = 3T$, $T_D = 12T$, $K_U = 1$, $K_D = 1$	Binary inverse-repeat
5	Process B: $T_U = 12T$, $T_D = 3T$, $K_U = 4$, $K_D = 1$	Binary inverse-repeat
6	Process A: $T_U = 3T$, $T_D = 12T$, $K_U = 4$, $K_D = 1$	Binary inverse-repeat
7	Process E: $T_U = 3T$, $T_D = 12T$, $K_U = 1$, $K_D = 1$	Five-level inverse-repeat
8	Process B: $T_U = 12T$, $T_D = 3T$, $K_U = 4$, $K_D = 1$	Five-level inverse-repeat
9	Process A: $T_U = 3T$, $T_D = 12T$, $K_U = 4$, $K_D = 1$	Five-level inverse-repeat
10	Process E: $T_U = 3T$, $T_D = 12T$, $K_U = 1$, $K_D = 1$	Multi-level (multisine)
11	Process B: $T_U = 12T$, $T_D = 3T$, $K_U = 4$, $K_D = 1$	Multi-level (multisine)
12	Process A: $T_U = 3T$, $T_D = 12T$, $K_U = 4$, $K_D = 1$	Multi-level (multisine)

Table 6.1. Summary of first order direction-dependent process parameters and perturbation signals.

For the neural network fitting, a separate set of training data (using a similar type of signal) was provided to train the network before using it to model the direction-dependent process. The final weights from the input layer to the hidden layer are summarised in Table 6.3. It is interesting to note that in Case 1, all the hidden neurons are independent of the values of $y(t\text{-}2)$ while this is true for two of the hidden neurons in Case 3. The reason behind this is that since the signal is binary, the direction of the output depends only on the input. The values of $y(t\text{-}2)$ are therefore theoretically not needed to provide the information on the output gradient. (The dependence of all the neurons on $y(t\text{-}2)$ in Case 2 may be due to some convergence problems in the network optimisation. In this chapter, all the neural networks were trained with random initial values set by the function *newff* and terminated when further improvement was small.)

The mean squared error in the outputs are given in Table 6.4. From the table, it can be seen that the neural network performs better than the Wiener model in all these cases

(Cases 1 to 3). However, both models are excellent as is clear from Figure 6.2, where the outputs for Case 3 are plotted.

Case	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T
1	0.59	0.64	0.59	0.57	4.79	6.35	7.16
2	0.03	1.68	0.96	0.60	4.96	6.55	7.00
3	2.97	1.60	0.94	0.82	4.78	6.44	7.96
4	0.59	0.66	0.64	0.60	4.88	7.22	8.89
5	0.03	1.64	1.01	0.81	4.88	7.22	8.90
6	2.97	1.64	1.01	0.81	4.88	7.22	8.90
7	0.74	0.90	0.16	0.02	6.01	4.05	10.05
8	-0.02	1.65	0.53	0.37	4.93	4.40	7.91
9	2.36	0.24	0.26	2.92	1.17	3.02	20.89
10	0.51	0.93	0.57	0.00	6.53	10.99	-
11	-0.12	1.63	0.66	0.59	5.17	3.57	4.30
12	2.24	2.22	1.28	-1.81	5.92	16.55	38.04

Table 6.2. Parameters of the Wiener model for first order processes.

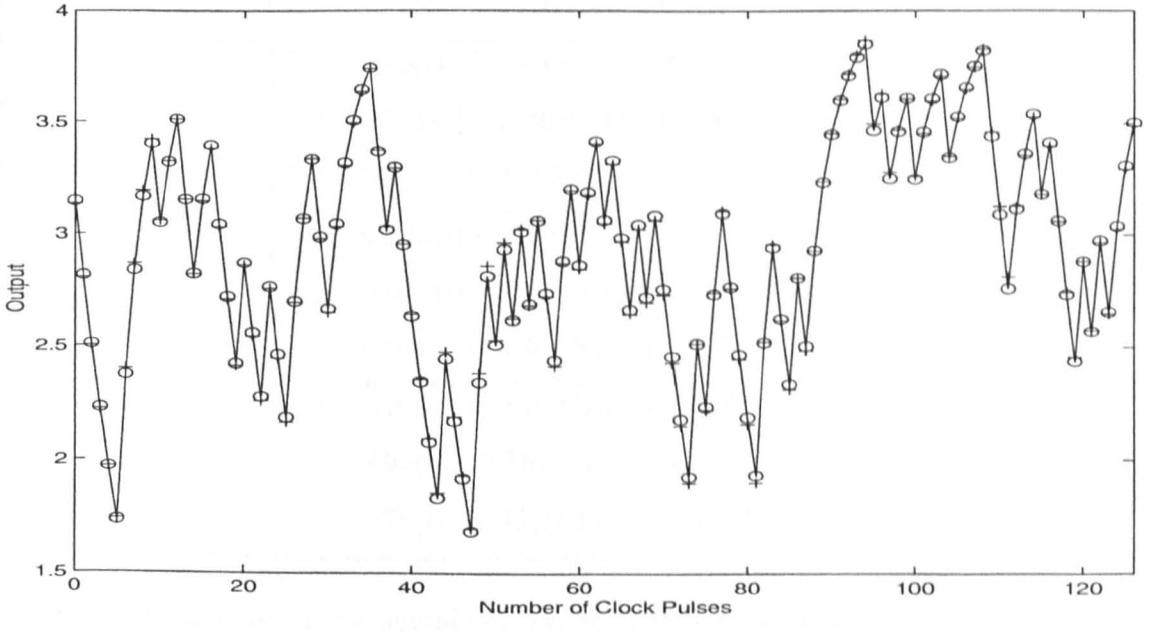

Figure 6.2. Outputs for Case 3. Solid line: Actual values; Plusses: Wiener model; Circles: Neural network model.

Case	Hidden layer parameters
1 and 4	$I_1 = 2.10u(t) - 0.46y(t\text{-}1) - 2.03$ $I_2 = -0.33u(t) + 0.46y(t\text{-}1) - 0.56$ $I_3 = -0.06u(t) + 0.22y(t\text{-}1) - 0.16$
2 and 5	$I_1 = 0.37u(t) - 0.01y(t\text{-}1) + 0.03y(t\text{-}2) - 0.38$ $I_2 = 0.18u(t) - 0.03y(t\text{-}1) + 0.02y(t\text{-}2) - 0.18$ $I_3 = -2.24u(t) + 0.29y(t\text{-}1) - 0.29y(t\text{-}2) - 2.80$
3 and 6	$I_1 = -0.09u(t) - 11.99y(t\text{-}1) + 8.82y(t\text{-}2) + 3.24$ $I_2 = 1.01u(t) + 0.05y(t\text{-}1) + 0.43$ $I_3 = 0.33u(t) - 0.04y(t\text{-}1) - 0.07$
7	$I_1 = -0.06u(t) - 0.15y(t\text{-}1) - 0.85y(t\text{-}2) + 1.85$ $I_2 = 0.02u(t) + 0.58y(t\text{-}1) - 0.19y(t\text{-}2) + 0.06$ $I_3 = -0.21u(t) - 1.76y(t\text{-}1) + 1.81y(t\text{-}2) - 0.24$
8	$I_1 = 0.01u(t) + 0.61y(t\text{-}1) - 0.96y(t\text{-}2) + 0.50$ $I_2 = -0.04y(t\text{-}1) - 0.04y(t\text{-}2) + 0.23$ $I_3 = 3.54u(t) + 0.02y(t\text{-}1) + 0.04y(t\text{-}2) + 3.36$
9	$I_1 = -0.09u(t) + 0.13y(t\text{-}1) - 0.20y(t\text{-}2) + 0.45$ $I_2 = -0.09u(t) + 0.39y(t\text{-}1) - 0.31y(t\text{-}2) - 0.20$ $I_3 = 0.27u(t) - 0.83y(t\text{-}1) + 1.04y(t\text{-}2) + 0.69$
10	$I_1 = 5.78u(t) - 2.73y(t\text{-}1) - 2.89y(t\text{-}2) - 2.42$ $I_2 = -0.01u(t) + 2.36y(t\text{-}1) - 1.85y(t\text{-}2) - 1.07$ $I_3 = -0.09y(t\text{-}1) + 0.35y(t\text{-}2) + 0.41$
11	$I_1 = -0.63u(t) - 0.84y(t\text{-}1) + 0.36y(t\text{-}2) + 0.52$ $I_2 = -0.01u(t) + 1.22y(t\text{-}1) - 1.00y(t\text{-}2) - 1.03$ $I_3 = -0.07y(t\text{-}1) + 0.17y(t\text{-}2) + 0.49$
12	$I_1 = -0.09y(t\text{-}1) + 0.02y(t\text{-}2) + 0.09$ $I_2 = 1.49u(t) + 2.76y(t\text{-}1) - 1.99y(t\text{-}2) - 5.26$ $I_3 = 7.37u(t) - 1.73y(t\text{-}1) + 0.50y(t\text{-}2) - 0.19$

Table 6.3. Parameters of the neural network hidden layer for first order processes.

Case	MSE, Wiener model	MSE, Neural network model
1	$5.51 * 10^{-5}$	$9.26 * 10^{-8}$
2	$4.25 * 10^{-5}$	$1.86 * 10^{-5}$
3	$2.47 * 10^{-4}$	$1.20 * 10^{-6}$
4	$8.47 * 10^{-5}$	$1.05 * 10^{-7}$
5	$4.98 * 10^{-4}$	$1.83 * 10^{-5}$
6	$4.98 * 10^{-4}$	$1.59 * 10^{-6}$
7	$1.05 * 10^{-2}$	$1.09 * 10^{-3}$
8	$3.50 * 10^{-3}$	$1.19 * 10^{-2}$
9	$5.19 * 10^{-1}$	$7.96 * 10^{-1}$
10	$5.04 * 10^{-3}$	$1.35 * 10^{-2}$
11	$1.45 * 10^{-1}$	$1.71 * 10^{-1}$
12	$3.42 * 10^{-1}$	$1.33 * 10^{0}$

Table 6.4. First order processes: Output mean squared error (MSE) for the two models.

In Cases 4 to 6, an inverse-repeat signal generated from the MLB signal used in Cases 1 to 3 was applied. For the Wiener model, the mean squared error in the outputs (shown in Table 6.4) was found to be larger than those obtained when the original signal was used. This could be due to the fact that the odd and even components were optimised separately. Despite this, the parameter estimates obtained using the inverse-repeat signal are more consistent, as can be seen from Table 6.2.

For the neural network model, the same parameter values were used as for Cases 1 to 3 since the inverse-repeat property was not utilised in any way. Hence, the magnitudes of the errors obtained are similar to those in Cases 1 to 3.

In Cases 7 to 9, the processes were perturbed with a five-level signal with $N = 124$ generated from GF(5) with the primitive polynomial $1 \oplus_5 4x \oplus_5 4x^2 \oplus_5 2x^3$. (This signal is an inverse-repeat signal.) The sampling interval was set to twenty times the clock-pulse interval in order to accurately calculate the output. The Wiener parameters

are shown in Table 6.2. The final weights from the input layer to the hidden layer are tabulated in Table 6.3. (These were again obtained with a separate set of training data.) From Table 6.3 (for Cases 7 to 9), all the hidden neurons (I_1, I_2 and I_3 in Figure 6.1) are now a function of $y(t\text{-}2)$. This is due to the fact that the sign of the slope of the output no longer depends only on the input, but also on the past values of the output which are essentially providing the output gradient information.

The mean squared error in the outputs are given in Table 6.4. From the table, it can be seen that the performance of the neural network is considerably worse for Cases 7 to 9 than it is for Cases 1 to 3 and 4 to 6. For the latter, the signal is binary and hence the slope of the output depends entirely on the input. Less information is required by the neural network to calculate the output and very high accuracy is achieved. The above is not true in Case 7 where the perturbation signal has five levels. However, the fit using the neural network is still very good, and it appears that this is because the gains in the two directions are equal (as is the case for most direction-dependent systems). The match using the Wiener model is also reasonably good. These are plotted in Figure 6.3.

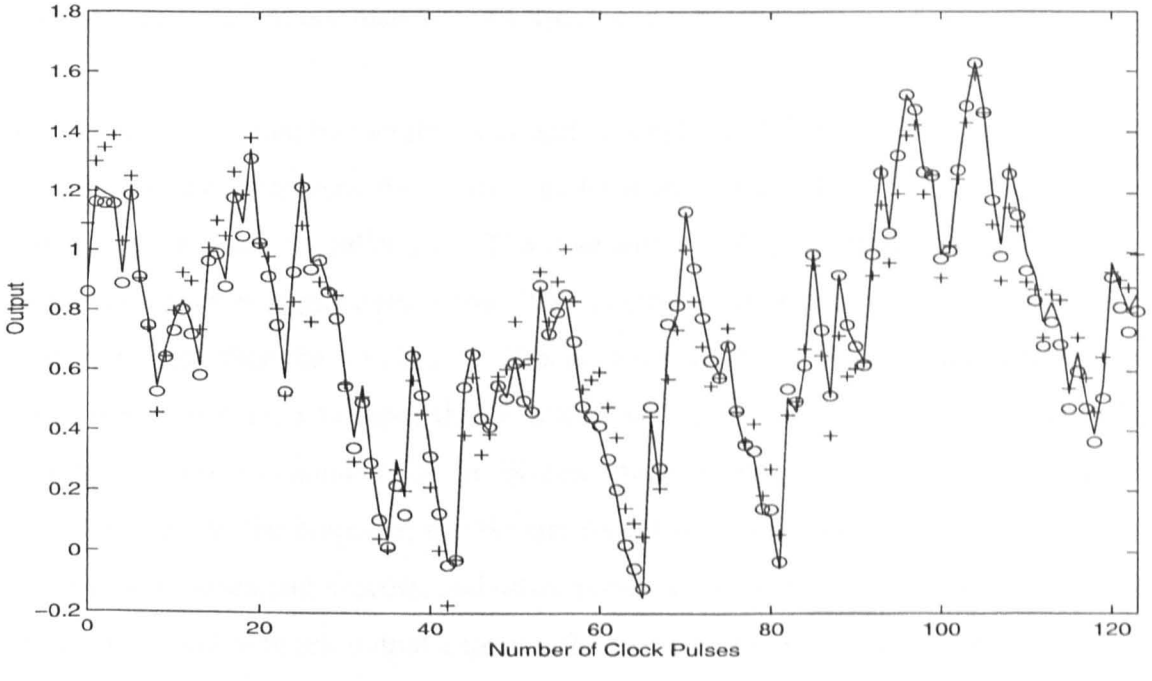

Figure 6.3. Outputs for Case 7. Solid line: Actual values; Plusses: Wiener model; Circles: Neural network model.

For Cases 8 and 9, the Wiener model performs better than the neural network. It was found that the neural network has difficulty in cases when the gains in the two directions of the actual process are different (except when the signal is binary, because then the different gains correspond to a change in the signal levels, with a one-to-one mapping). This causes the output waveforms to be increasingly complicated to model. These are illustrated in Figures 6.4 and 6.5 (top) for Cases 8 and 9 respectively. An important feature to note is the symmetry in the outputs. In Figure 6.4, the output is much more symmetrical about zero than that in Figure 6.5. This is because the direction with a larger gain has slower dynamics which compensates to some extent for the larger gain. Thus the percentage of the time when the output increases is not very different from that when it decreases and hence, the nonlinear effects are not so apparent. This can also be seen in Figures 6.4 and 6.5 (middle) where in Figure 6.4, the odd order terms (which comprise mainly of the linear term) dominate while this is untrue in Figure 6.5. The magnitude of the output discrete Fourier transform of the two processes are shown in the bottom of Figures 6.4 and 6.5. These plots again confirm that the departure from linearity is very much more pronounced in Case 9 and hence explains the fact that for the optimised Wiener parameters, $K_3 > K_2 > K_1$. (Also, the amount of nonlinearity generated in Case 7 is between that of Cases 8 and 9.)

It is interesting to note that while the neural network has difficulty coping with different gains in the two directions, the Wiener model works better when the direction with the faster dynamics has a smaller gain. The case with equal gains in both directions is not preferred, which is slightly surprising. This is clear from Table 6.4, where the error in Case 7 is larger than that in Case 8. (This is also true with a binary signal perturbation, and the error in Case 1 is larger than that in Case 2.) Another point worth noting is that although the error is smaller for the Wiener model compared with the neural network model in Case 9, the output of the Wiener model does not seem to follow the trend of the direction-dependent process, and often moves in the opposite direction. On the other hand, the neural network output captures the trend of the actual process better, but has an offset to the true values during most of the period.

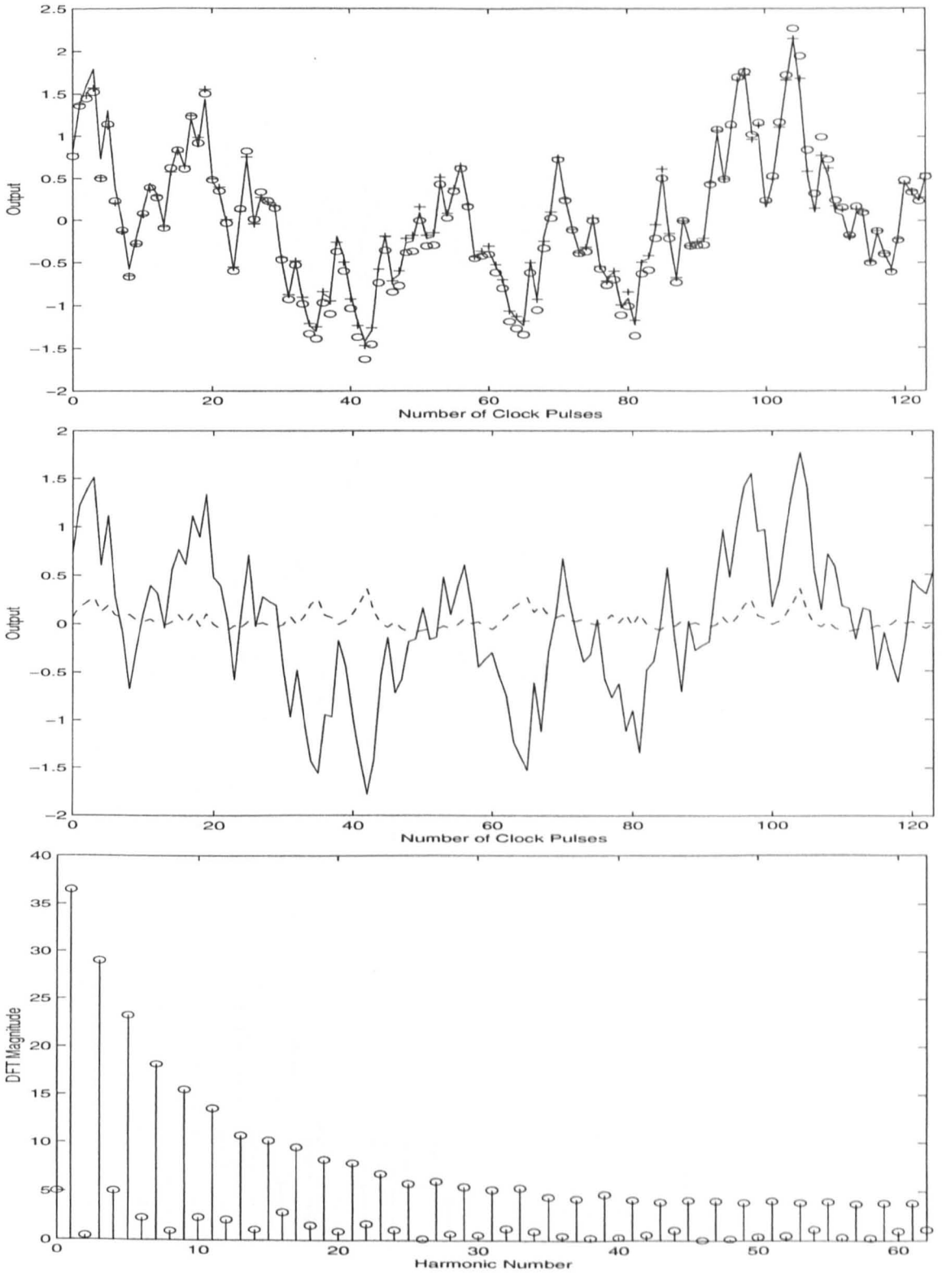

Figure 6.4. Case 8.
Top: Process and model outputs. Solid line: Actual values; Plusses: Wiener model; Circles: Neural network model.
Middle: Solid line: Odd order terms; Dashed line: Even order terms.
Bottom: Discrete Fourier transform magnitude of the process output.

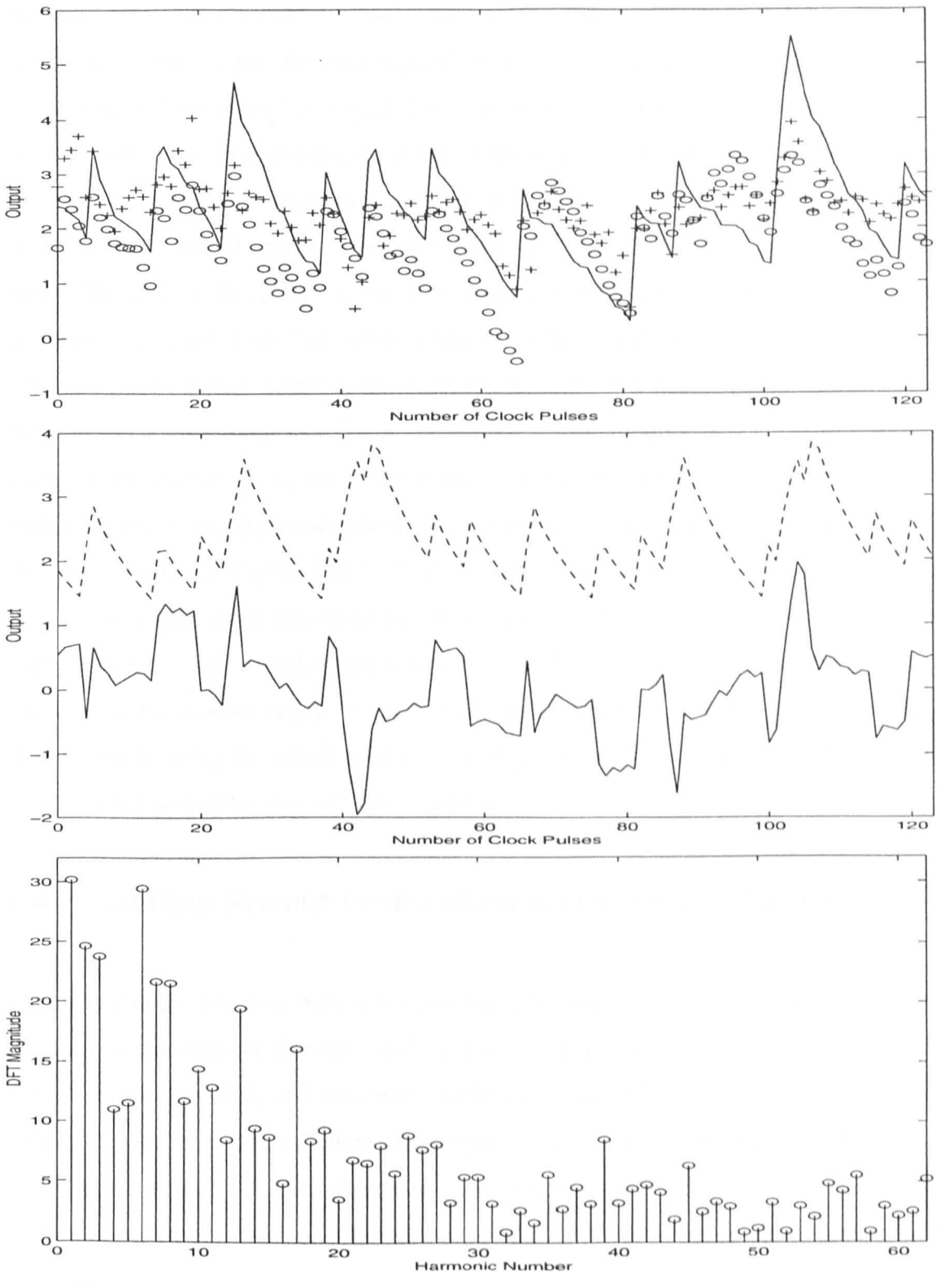

Figure 6.5. Case 9.
Top: Process and model outputs. Solid line: Actual values; Plusses: Wiener model; Circles: Neural network model.
Middle: Solid line: Odd order terms; Dashed line: Even order terms.
Bottom: Discrete Fourier transform magnitude of the process output.

In Cases 10 to 12, a multisine signal with 16 consecutive harmonics and $N = 256$ was used. The outputs of the direction-dependent process, the Wiener model and the neural network model are plotted in Figure 6.6. The error obtained using the Wiener model is smaller than that obtained using the neural network model for all these cases, and these are given in Table 6.4. The better performance of the Wiener model compared to the neural network model appears to be due to the fact that the input is a smoothly varying signal. This causes the change in the dynamics to be more gradual and happen at longer intervals compared with that when using a pseudo-random signal. For the neural network, this particular nature of the input signal causes one of the hidden neurons to be independent of the input $u(t)$. Thus the current output is more dependent on the past values of the output $y(t\text{-}1)$ and $y(t\text{-}2)$ than on the current input, which can be expected with a smoothly varying signal. The match using the Wiener model is very good in Case 10, and less good in Cases 11 and 12. (For a smoothly varying signal, a better match is obtained when the gains in both directions are equal, and not when the direction with the faster dynamics has a smaller gain.) This may be due to the fact that the larger gain in the upward direction causes the positive peaks to be sharper, and hence more difficult to model. The fit using the neural network model is reasonable in Cases 10 and 11 (except for the initial part of the period), but is poor in Case 12.

6.4 Modelling Second Order Direction-dependent Process

For second order direction-dependent processes, the sign of the slope of the output does not change immediately following a change in the input. These processes are inherently very difficult to model, and the neural network model in Figure 6.1 was found to be unable to cope with the modelling of such processes; the output in all the cases studied so far bore almost no resemblance to that of the direction-dependent process. This section therefore considers only the performance of the Wiener model in modelling such processes.

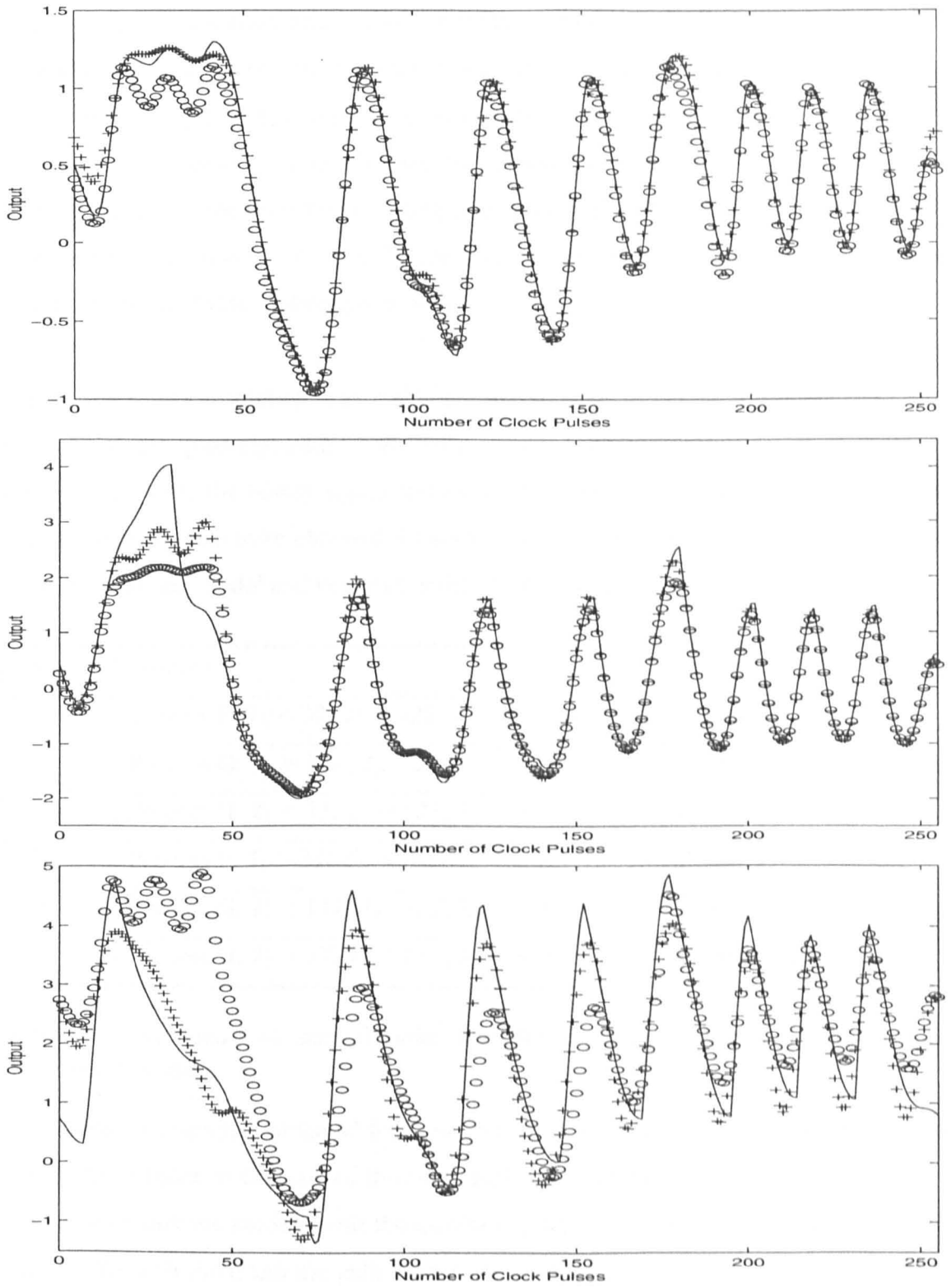

Figure 6.6. Process and model outputs. Solid line: Actual values; Plusses: Wiener model; Circles: Neural network model.
Top: Case 10.
Middle: Case 11.
Bottom: Case 12.

The three processes considered were the same as Processes E, B and A in Table 6.1, except that in each case, the denominators of the transfer functions in each direction were squared. (These have been renamed as Processes F, G and H respectively and are summarised in Table 6.5.) The Wiener model has the structure shown in Figure 5.6. Critical damping was used for the linear, quadratic and cubic paths as this restricts the number of parameters to optimise, thus preventing convergence problems. An additional advantage is the shorter optimisation time.

Since no theoretical solution is available for second order direction-dependent processes, the sign of the quadratic path is unknown, which further complicates the analysis. In view of this, only the binary signal and its inverse-repeat were used in the simulations. Two Wiener models were obtained for each case with the sign of the quadratic path set to positive in one model and negative in the other. The results obtained were compared.

Case	Process	Input Signal
13	Process F: $T_U = 3T$, $T_D = 12T$, $K_U = 1$, $K_D = 1$	Binary
14	Process G: $T_U = 12T$, $T_D = 3T$, $K_U = 4$, $K_D = 1$	Binary
15	Process H: $T_U = 3T$, $T_D = 12T$, $K_U = 4$, $K_D = 1$	Binary
16	Process F: $T_U = 3T$, $T_D = 12T$, $K_U = 1$, $K_D = 1$	Binary inverse-repeat
17	Process G: $T_U = 12T$, $T_D = 3T$, $K_U = 4$, $K_D = 1$	Binary inverse-repeat
18	Process H: $T_U = 3T$, $T_D = 12T$, $K_U = 4$, $K_D = 1$	Binary inverse-repeat

Table 6.5. Summary of second order direction-dependent process parameters and perturbation signals.

The Wiener parameters obtained for Cases 13 to 16 are given in Table 6.6. In all these cases, the estimate of the gain of the cubic path was found to be zero. From the table, it can be seen that the models with the quadratic path added have small values of T_2 and the models with the quadratic path subtracted have large values of T_2. The values of T_1 lie between these values of T_2. The errors obtained are tabulated in Table 6.7 (the modified Wiener model will be discussed later in the section) and these are smaller for the models with the sign of the quadratic path being positive. This is slightly surprising

since the dynamics are faster in the upward direction in Cases 13 and 15, but slower in the direction mentioned in Case 14. The sign of the quadratic path which gives a lower error was expected to be different in Cases 13 and 15 compared with that in Case 14. (For a first order process, if the sign of the quadratic path is incorrect, the optimised gain of that path will be zero.) The graphs of the process and model outputs are shown in Figure 6.7. Again, the fit is better for Case 14 than for Case 13, despite the larger error in Case 14 which is due to the fact that the overall gain is larger in Case 14. The better performance of the Wiener model when the dynamics are faster in the direction with a smaller gain (when a pseudo-random signal is used) is consistent with the results obtained earlier in Section 6.3.

Case	Sign	K_0	K_1	K_2	T_1/T	T_2/T
13	+	0.10	1.05	0.71	6.76	2.51
13	-	0.18	1.25	1.50	6.87	11.41
14	+	-0.02	1.73	1.45	5.67	7.02
14	-	0.05	2.06	1.38	6.16	18.12
15	+	0.41	2.30	1.81	5.50	3.11
15	-	0.72	2.92	2.89	5.69	15.56

Table 6.6. Parameters of the Wiener model for second order processes using an MLB input.

Case	MSE with quadratic path added	MSE with quadratic path subtracted	MSE with modified Wiener model
13	$3.86 * 10^{-3}$	$5.48 * 10^{-3}$	$3.03 * 10^{-3}$
14	$4.37 * 10^{-3}$	$8.20 * 10^{-3}$	$1.85 * 10^{-3}$
15	$7.92 * 10^{-2}$	$1.35 * 10^{-1}$	$7.44 * 10^{-2}$

Table 6.7. Second order processes: Output mean squared error (MSE) for the Wiener model using an MLB input.

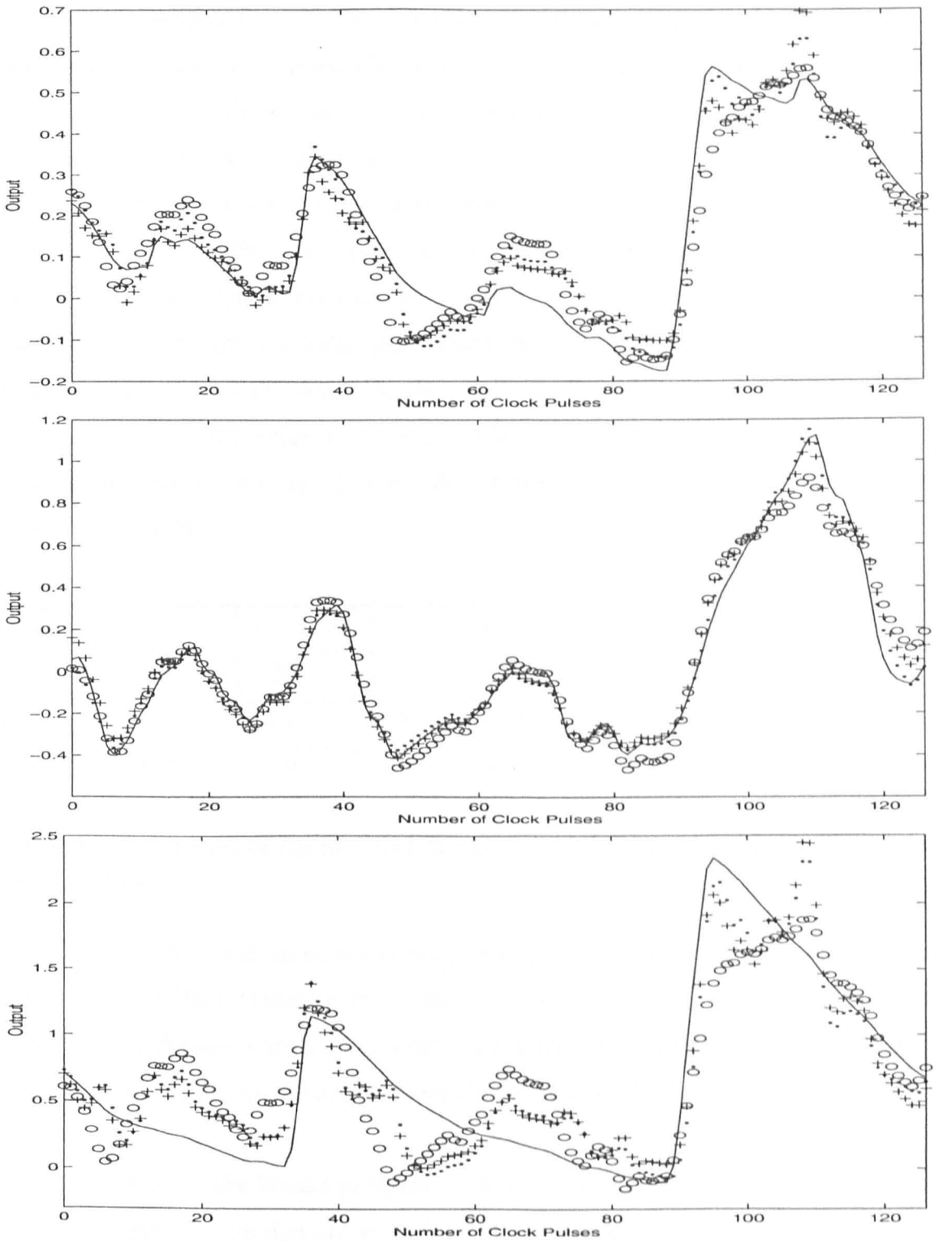

Figure 6.7. Process and model outputs. Solid line: Actual values; Plusses: Wiener model with quadratic path added; Circles: Wiener model with quadratic path subtracted; Dots: Modified Wiener model.
Top: Case 13.
Middle: Case 14.
Bottom: Case 15.

To further investigate the use of the Wiener model in modelling second order direction-dependent processes, a modified Wiener model with two quadratic paths (one added and the other subtracted) was used. A better match was obtained for all the three cases considered. The Wiener parameters obtained using this model are given in Table 6.8 and the error measurements are tabulated in Table 6.7. From Table 6.8, the values for T_{2+} (for the quadratic path with a positive sign) and T_{2-} (for the quadratic path with a negative sign) are now in between the values of T_2 (those obtained with the original models, one with the quadratic path added and the other with the quadratic path subtracted) shown in Table 6.6. Also, the values for T_{2+} and T_{2-} are closer to the smaller values of T_2 in the original Wiener model. The improvement brought by the modification can be seen in Table 6.7 though the degree of it varies across the three cases investigated.

Case	K_0	K_1	K_{2+}	K_{2-}	T_1/T	T_{2+}/T	T_{2-}/T
13	0.13	1.20	2.02	2.12	7.05	4.30	5.10
14	-0.01	1.67	8.38	8.52	5.78	10.95	11.47
15	0.46	2.60	6.14	6.05	5.90	4.61	4.94

Table 6.8. Parameters of the modified Wiener model for second order processes using an MLB input.

From Figure 6.7, it can be seen that the match in Case 13 is good while the match in Case 14 is excellent. However, the quality of fit is less good in Case 15. It should be noted that the dynamics are markedly different in the two directions in this case, and for most practical situations, the dynamics would not be so different from one another.

For Cases 16 to 18, the Wiener parameters and the mean squared error in the outputs are given in Table 6.9. The sign of the quadratic path is positive in Cases 16 and 18. (When the sign was set to negative, the estimate of K_2 was zero.) This path could not be estimated in Case 17. However, the cubic path could now be estimated due to the fact that the odd order components had been separated at the output before applying the optimisation. (The modified Wiener model was not used in Cases 16 to 18 because it

brought no noticeable improvement to the model and convergence problems were encountered during most of the simulations conducted.)

Case	K_0	K_1	K_2	K_3	T_1/T	T_2/T	T_3/T	MSE
16	0.14	1.07	0.41	0.01	7.24	2.26	8.05	$3.49 * 10^{-3}$
17	0.04	1.93	0.00	1.14	6.29	-	10.93	$6.37 * 10^{-3}$
18	0.48	2.23	0.34	0.01	6.94	5.14	8.11	$1.01 * 10^{-1}$

Table 6.9. Parameters of the Wiener model for second order processes and the output mean squared error (MSE) using an inverse-repeat MLB input.

6.5 Practical Application using Electronic Nose

6.5.1 Model of the Gas Sensor

The gas sensor in the electronic nose [76 - 79] can be modelled using the adsorption-desorption reaction described by

$$AS + \{\ \} \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \{AS\} \tag{6.2}$$

where AS is the adsorbed species, { } represents an empty adsorption site, $\{AS\}$ represents an occupied site, k_f is the forward rate constant and k_b is the backward rate constant. Denoting the fractional occupancy of adsorption sites and the concentration of the target gas by θ and C respectively gives

$$\dot{\theta} = k_f C - (k_b + k_f C)\theta \tag{6.3}$$

Assuming that the experiment is started at time $t = 0$, and the sensor is in steady-state in air at this time, so that $\theta(0) = 0$, the solution to (6.3) is

$$\theta(t) = \frac{KC}{1+KC}\left(1 - \exp\left\{-(k_b + k_f C)t\right\}\right) \tag{6.4}$$

where $K = \frac{k_f}{k_b}$. The conductance G of the sensor can be written as

$$G(t) = G(0) + a\Gamma\theta(t) \tag{6.5}$$

where $G(0)$ is the conductance of the sensor in air, a is the sensitivity coefficient and Γ is the concentration of the adsorption sites. Substituting (6.4) into (6.5) gives

$$\Delta G(t) = G(t) - G(0) = a\Gamma\left(\frac{KC}{1+KC}\left(1-\exp\left\{-(k_b + k_f C)t\right\}\right)\right) \tag{6.6}$$

Similar analysis can be done for the case when the target gas is removed and the sensor is again placed in air. The direction-dependent characteristics are caused by C having different values in the upward and downward directions. Also, it should be noted that the dynamics are predominantly first order (from the exponential term in equation (6.6)); this may not be exact as the equation is from that of a simplified model.

6.5.2 Experimental Setup

The experiment was carried out using the software LabVIEW [80, 81]. (The program was written by Graham Searle of the University of Warwick.) The input to the program is the position of the valves through which chemical odours can reach the metal oxide semiconductor (MOS) sensor. In this particular experiment, acetone was used as the positive input, and air as the negative input. The output of the program is the voltage across the sensor, which is inversely proportional to the conductance given in (6.5).

6.5.3 Detection and Estimation of Direction-dependent Dynamics

An MLB signal of length 63 with shift register feedback from stages 1, 4, 5 and 6 (Signal 1) was applied as input. This signal was used because the second order terms due to the shift-and-add property (refer to Section 2.3.1.2) are furthest from the zero-lag position compared with other MLB signals of the same length [29], hence enabling easy detection of these terms in the crosscorrelation function. The resulting second order and third order shifts are given in Table 6.10. The output was sampled at every second while the clock-pulse interval was set to ten seconds. There was a time delay of approximately three seconds and this was removed before any analysis was conducted. This delay was due to the fact that after a valve is opened, the odour will take some time to reach the sensor. The output of the system is plotted in Figure 6.8 where it is evident that the

dynamics in the upward direction are slower than those in the downward direction. In other words, the adsorption of acetone on the MOS sensor surface is faster than its desorption, which is expected from the theory outlined in Section 6.5.1. The input-output crosscorrelation function, with the mean removed, is plotted in Figure 6.9 (top). (It is usual practice to remove the mean in the output before carrying out any identification.) The direction-dependent characteristic is confirmed here where coherent peaks are observed at the lags shown in Table 6.10. A smoother crosscorrelation function is obtained using the corresponding inverse-repeat signal (as illustrated in Figure 6.9 (bottom)) since the even order peaks are no longer present.

Shifts	k_1	k_2	k_3	k_4	k_5	m_1	m_2
Signal 1	39	15	11	30	28	35	47, 57
Signal 2	8	16	53	32	38	45	13, 41

Table 6.10. Resulting shifts due to second order terms k and third order terms m for two 63-digit MLB signals, one with feedback from stages 1, 4, 5 and 6 (Signal 1), and the other with feedback from stages 2, 3, 5 and 6 (Signal 2, used later in Section 6.5.4).

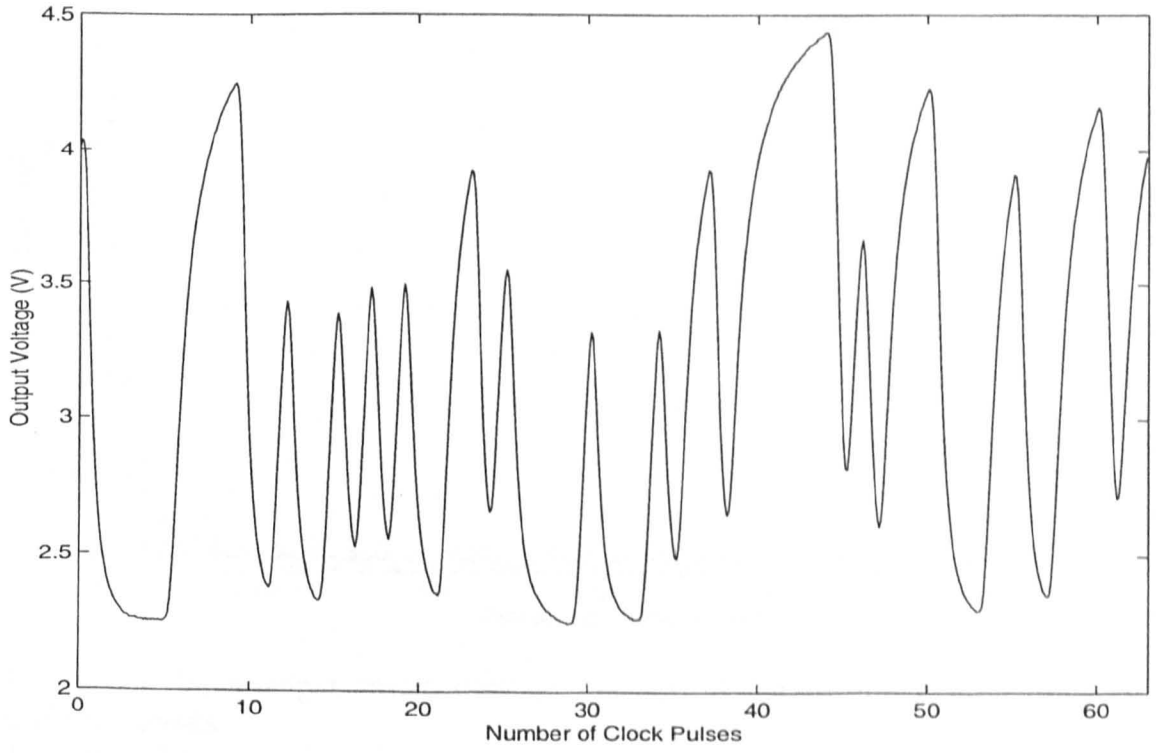

Figure 6.8. Output of the electronic nose using Signal 1.

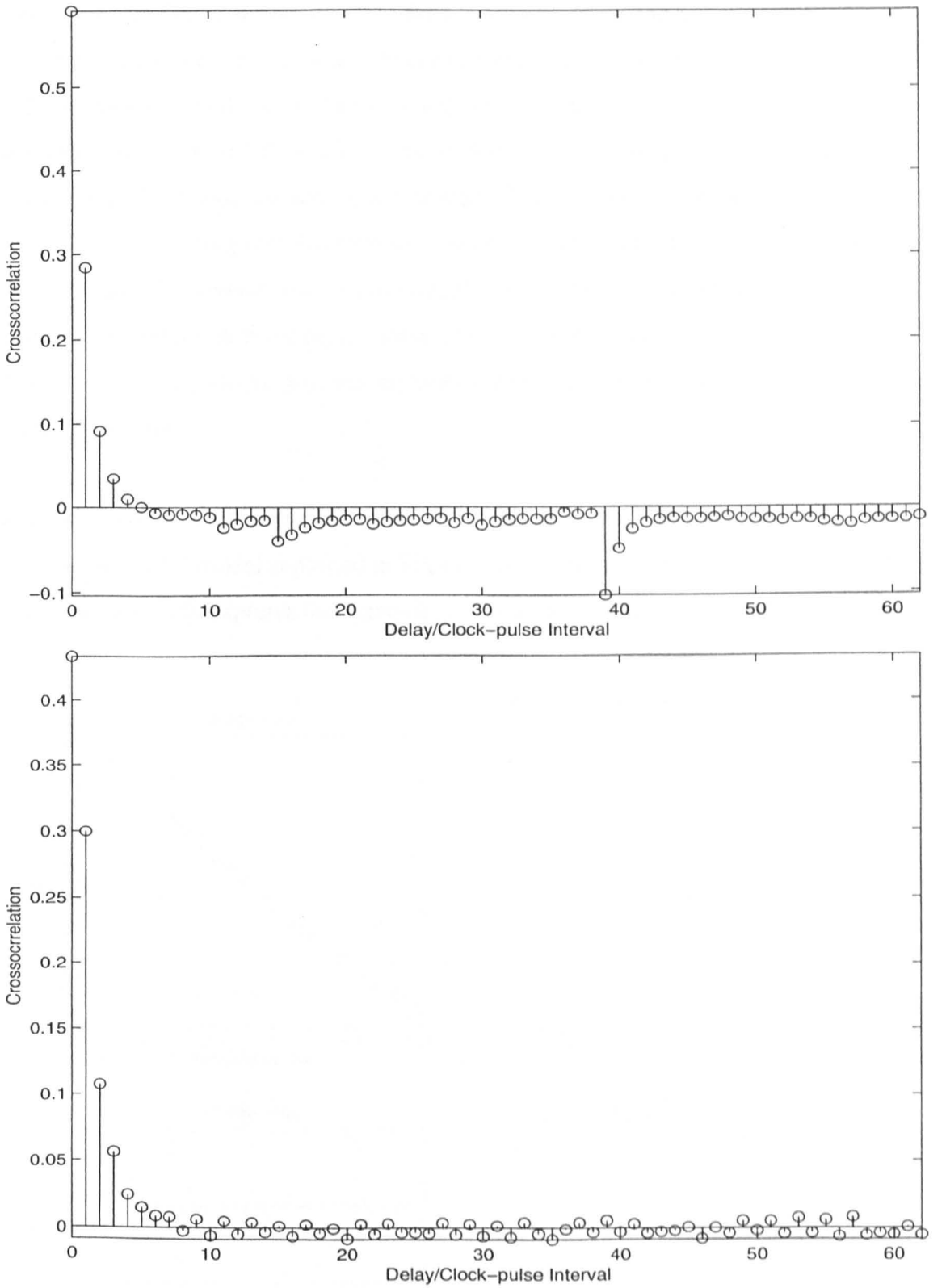

Figure 6.9. Input-output crosscorrelation function of the electronic nose using two different signals.
Top : Signal 1.
Bottom : Inverse-repeat of Signal 1.

The combined linear dynamics were then estimated (with the techniques discussed in Chapter 4) using both Signal 1 and its corresponding inverse-repeat signal. Both 0/1 and 1/2 models were tried and the results are given in Table 6.11. The time constants in the upward and downward directions were 18.49s and 5.54s respectively and these were obtained through separate step response tests. The theoretical combined time constant was 10.15s, assuming that the process was first order. From Table 6.11, it was found that although the process was predominantly first order, second order dynamics had a considerable effect on the system output. Hence, the theoretical combined time constant was not very accurate as this was calculated with the assumption that the process was strictly first order.

For the purpose of illustration, the result obtained for the inverse-repeat signal using ELiS with the 1/2 model is plotted in Figure 6.10. It can be seen from this figure that the model successfully captures the dynamics of the actual process.

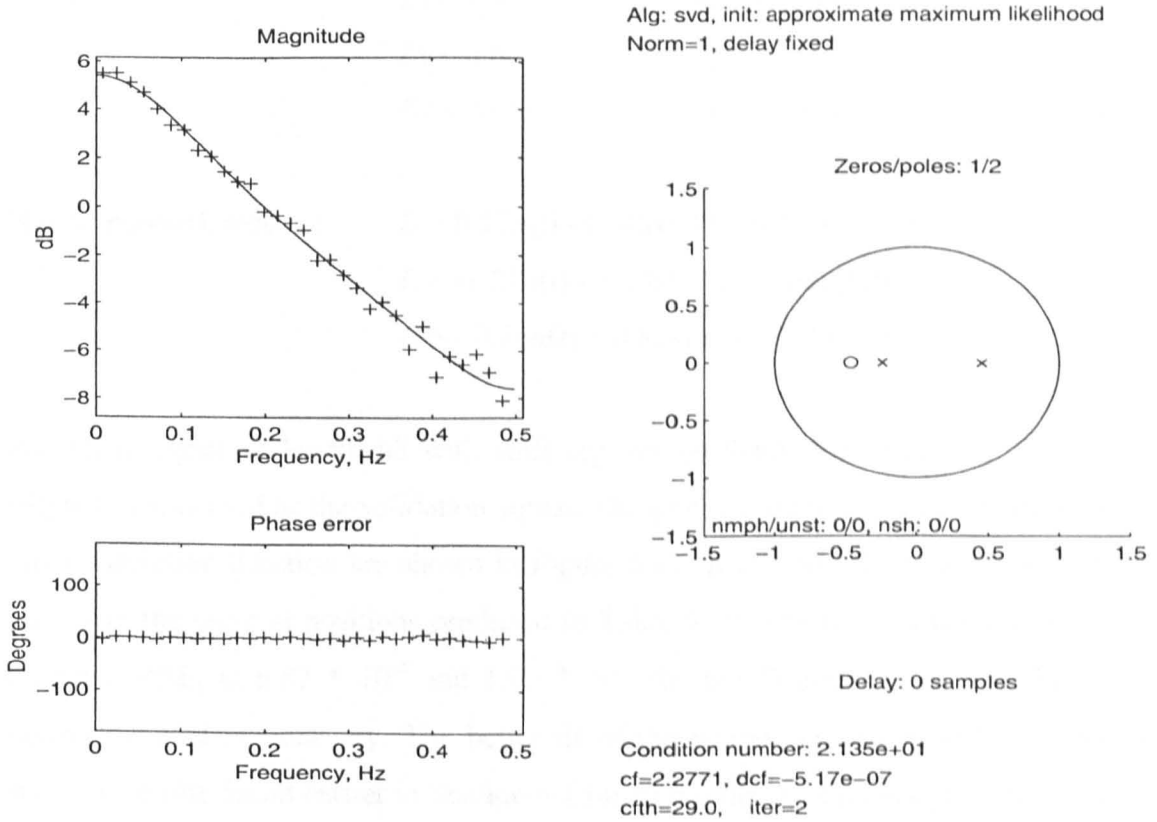

Figure 6.10. Result for ELiS using the inverse-repeat signal of Signal 1 and with a 1/2 model.

Model/ Algorithm	Signal 1 Order 0/1, T_C	Signal 1 Order 1/2, T_C	Inverse-repeat Order 0/1, T_C	Inverse-repeat Order 1/2, T_C
ARX	13.39	10.51	18.13	12.06
ARMAX	13.39	10.51	18.17	12.02
State-space	11.72	9.50, 55.38	11.27	5.08, 17.12
ELiS	12.90	12.67, 40.35	15.19	15.19

Table 6.11. Estimated combined time constant T_C. For models with two poles, only the positive poles were considered.

6.5.4 Modelling of Direction-dependent Dynamics

Signal 1 was used as the training signal to obtain optimised parameters for both the Wiener model and the neural network model. These values are given below :

Wiener model :

$$K_0 = -0.08$$
$$K_1 = 1.47 \qquad T_1 = 19.34\text{s}$$
$$K_2 = 0.66 \qquad T_2 = 25.82\text{s}$$
$$K_3 = -0.86 \qquad T_3 = 30.00\text{s} \qquad (6.7)$$

Neural network model :

$$I_1 = 0.17u(t) - 0.60y(t\text{-}1) + 0.13y(t\text{-}2) - 0.52$$
$$I_2 = -1.27u(t) - 5.79y(t\text{-}1) + 5.08y(t\text{-}2) - 0.18$$
$$I_3 = -0.26u(t) + 0.88y(t\text{-}1) + 0.25y(t\text{-}2) - 1.29 \qquad (6.8)$$

An MLB signal of length 63 with shift register feedback from stages 2, 3, 5 and 6 (Signal 2) was used as the validation signal. The plots of the output and the input-output crosscorrelation function are shown in Figure 6.11. It can be seen that coherent peaks appear in the latter at positions predicted in Table 6.10. The mean squared error in the outputs, MSE, is $6.87 * 10^{-2}$ and $1.09 * 10^{-2}$ for the Wiener model and the neural network model respectively. The better fit of the neural network model is consistent with the results found earlier in Section 6.3 for first order direction-dependent processes perturbed using binary signals. However, both models are seen to be very good.

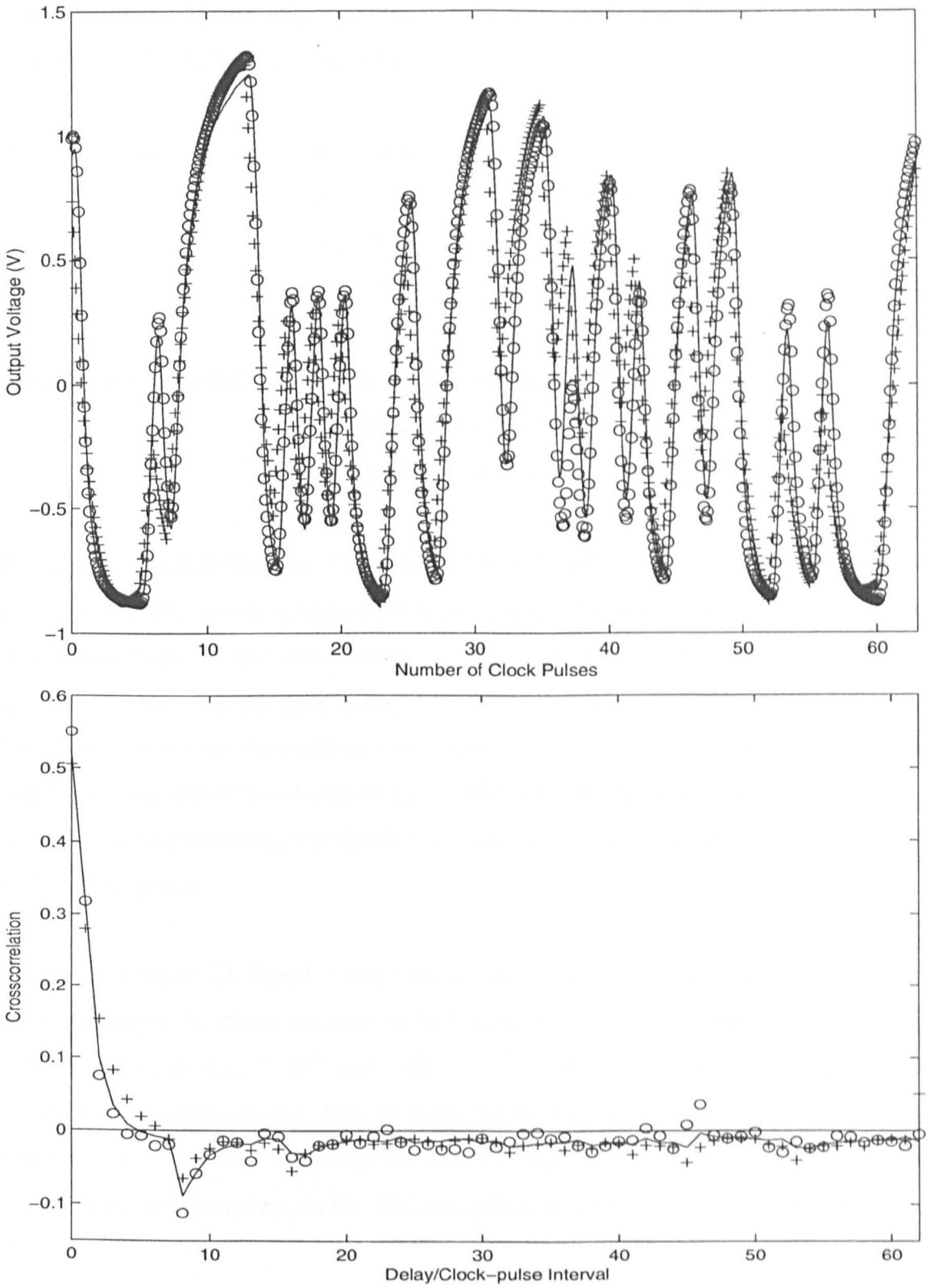

Figure 6.11. Modelling of the electronic nose using Signal 2. Solid line: Actual values; Plusses: Wiener model; Circles: Neural network model.
Top : Outputs.
Bottom : Input-output crosscorrelation functions.

The experiment was repeated using the inverse-repeat of Signal 1 as the training signal. The optimised parameters are given below :

Wiener model :

$$K_0 = -0.03$$
$$K_1 = 1.20 \qquad T_1 = 20.21\text{s}$$
$$K_2 = 0.56 \qquad T_2 = 43.04\text{s}$$
$$K_3 = -0.66 \qquad T_3 = 23.27\text{s} \qquad (6.9)$$

Neural network model :

$$I_1 = 1.72u(t) + 4.81y(t\text{-}1) - 4.16y(t\text{-}2) - 0.21$$
$$I_2 = 0.13u(t) - 0.10y(t\text{-}1) + 0.01y(t\text{-}2) - 0.03$$
$$I_3 = 1.69u(t) + 4.30y(t\text{-}1) - 3.85y(t\text{-}2) + 0.33 \qquad (6.10)$$

It should be noted that the electronic nose is a sensitive device and its dynamic characteristic changes noticeably with room temperature and humidity. The gain of the system was found to have decreased when this set of experiments was carried out. This was largely due to the decrease in the concentration of acetone as some of the molecules had dispersed into the surrounding atmosphere. In fact, the training was repeated for the neural network due to the change in gain. Otherwise, the parameters would be the same as when the non inverse-repeat signal was used since the inverse-repeat property was not utilised in any way.

The inverse-repeat of Signal 2 was used as the validation signal. The outputs and the crosscorrelation functions are shown in Figure 6.12. The mean squared error in the outputs, MSE, is $4.27 * 10^{-2}$ and $1.91 * 10^{-2}$ for the Wiener model and the neural network model respectively. The fit using the neural network is still better than that using the Wiener model. However, the match using the latter has improved due to the signal being inverse-repeat, as the odd and even orders terms were first separated in the output.

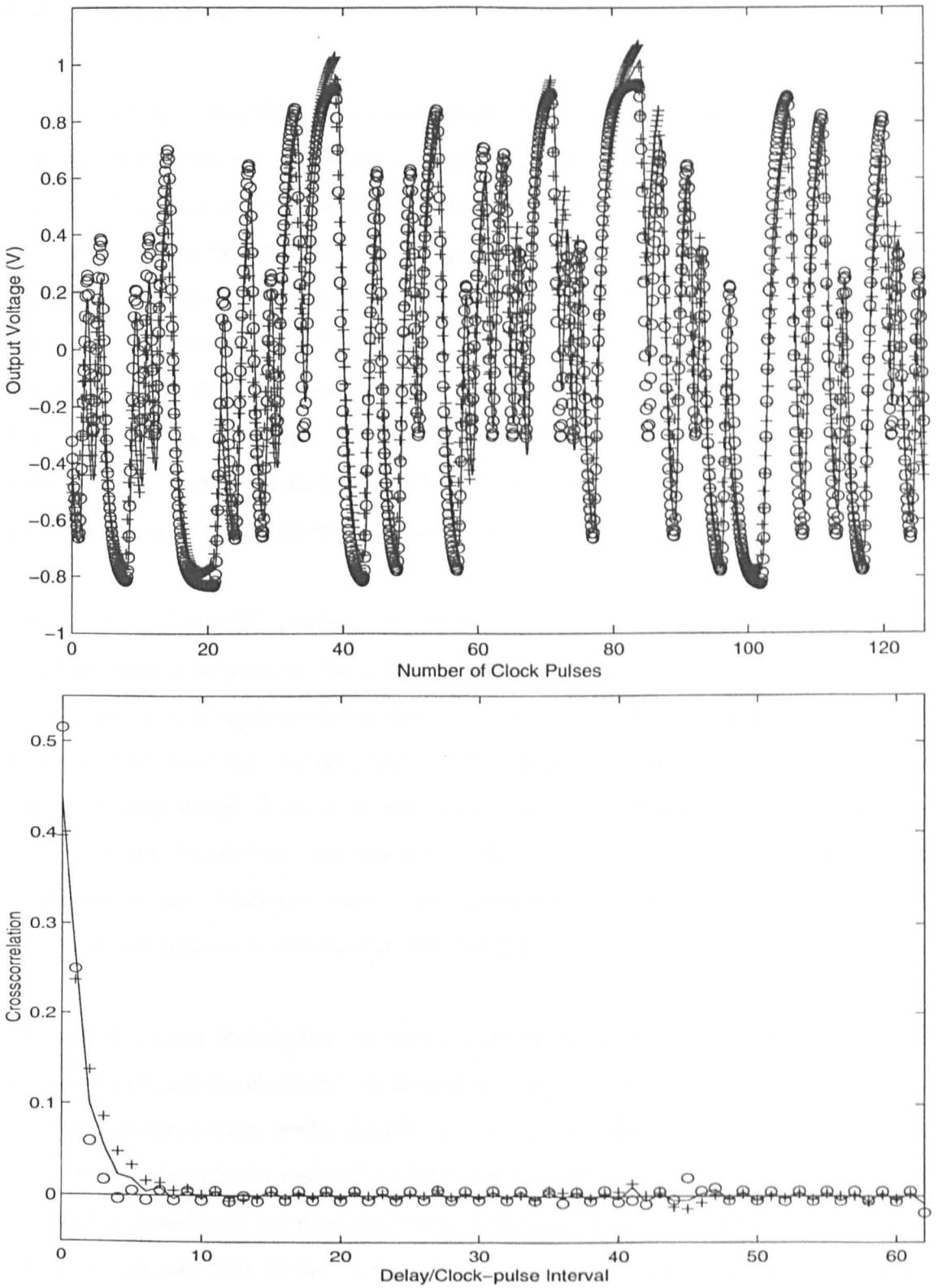

Figure 6.12. Modelling of the electronic nose using the inverse-repeat of Signal 2. Solid line: Actual values; Plusses: Wiener model; Circles: Neural network model.
Top : Outputs.
Bottom : Input-output crosscorrelation functions.

6.6 Conclusions

The use of Wiener models and neural networks for modelling processes with direction-dependent dynamics was considered. It was found that both types of model can be used when the dynamics in both directions are first order. For such a process with binary and inverse-repeat binary input signals, the agreement between the model and the process output is excellent for both types of model, but it should be noted that this is the only case for which the output of the process itself can be determined theoretically. For a five-level signal, the performance of the two models investigated are about the same. The worst results are obtained when the gain is larger in the direction with faster dynamics; this causes the nonlinear effects to be more apparent. For a multisine input signal, the Wiener model gives somewhat better results than the neural network.

It was not found possible to obtain any agreement between the outputs of a second order direction-dependent process and a neural network with the architecture shown in Figure 6.1. For binary and inverse-repeat binary input signals, a Wiener model still gave good agreement between the outputs, particularly when the steady-state gains in the two directions were equal. When a binary signal was used, a structure with two quadratic paths, one added to and one subtracted from the outputs of the constant and linear paths, was found to give closer agreement than models with a single quadratic path. However, the above was untrue for inverse-repeat binary input signals.

The modelling was then carried out on an electronic nose, where the input to the process is the strength of the chemical odour and the output is the voltage across the sensor. Since the dynamics are predominantly first order, and the input is binary, the neural network provides a better match than the Wiener model. However, both models are very good. This experiment not only illustrates the applicability of the modelling techniques involved, but also that of the methods used in the detection and estimation of linear dynamics which were covered in Chapters 3 and 4 respectively.

The overall conclusion is that Wiener models can be used to model direction-dependent processes with both first and second order dynamics, and that a neural network provides a useful alternative for first order processes.

Chapter 7

Control of Processes with Direction-dependent Dynamics

7.1 Introduction

In the first part of the chapter, the control of first order and second order processes with direction-dependent dynamics but with the same gain in both directions is examined. The Proportional-Integral-Derivative (PID) controller [82 - 87] is applied as it is by far the most dominating form of feedback in use today, with 90% of all control loops being PID [88]. In the industry, most direction-dependent processes are controlled using gain scheduling with two sets of controller parameters, one for each operating mode (direction of the output). An example in the literature is given in [89] where a heating/cooling process, namely a furnace, is considered. In [89], the direction-dependent characteristic is termed 'unsymmetrical', and it noted that the heating dynamics are normally much faster than the cooling dynamics, especially when no active cooling is provided.

In this chapter, the control of such processes using only one set of controller parameters, obtained from the combined linear dynamics of the process (which were discussed is Chapter 4), is investigated. The danger of not recognising the direction-dependent characteristics is implicitly shown though simulation examples, where very poor controller performance, or even instability, is encountered in some of these examples. Thus, the importance of the detection of the departure from linearity (see Chapter 3) is emphasised.

Two alternative implementations of the derivative term are first used, and the controller with the best overall response (in terms of the smoothness and the speed of response) is then chosen. The effects of varying the controller gain and the controller period (which will be defined later in the chapter) of the closed-loop system are studied for this controller. This chapter then moves on to look at the control of first order and second order processes with different gains in the positive and negative directions. The problems that arise are discussed and modifications in the controller algorithm for possible improvements are suggested.

7.2 Control of Processes with the Same Gain in Both Directions

7.2.1 Control of First Order Process

7.2.1.1 Control Using PID

The structure of the control loop is shown in Figure 7.1 where G_{CO} is the controller, G_{PR} is the process, r is the setpoint, e is the error signal, y is the output of the controller and c is the controlled variable.

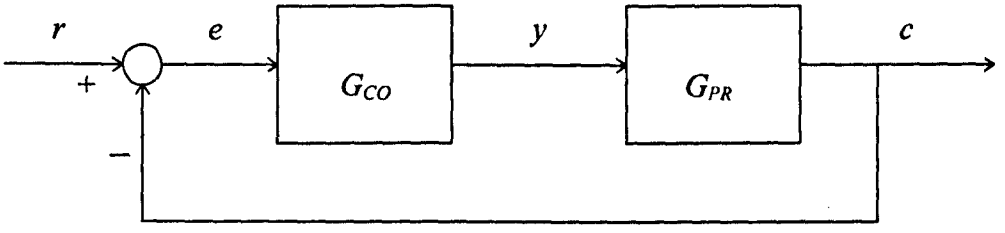

Figure 7.1. Structure of the control loop.

The formula for the PID controller is

$$y = G_C\left(e + \frac{1}{\tau_I}\int e dt + \tau_D \frac{de}{dt}\right) \tag{7.1A}$$

where G_C is the controller gain, τ_I is the integral time constant and τ_D is the derivative time constant. Differentiating with respect to e gives

$$\frac{dy}{dt} = G_C\left(\frac{de}{dt} + \frac{e}{\tau_I} + \tau_D \frac{d^2e}{dt^2}\right) \tag{7.1B}$$

Its discrete approximation is implemented using MATLAB

$$y_n = y_{n-1} + G_C\left[\left(1 + \frac{T}{\tau_I} + \frac{\tau_D}{T}\right)e_n - \left(1 + \frac{2\tau_D}{T}\right)e_{n-1} + \frac{\tau_D}{T}e_{n-2}\right] \tag{7.2}$$

where T is the sampling interval.

The values for G_C, τ_I and τ_D according to the Ziegler-Nichols settings [90] are

$$G_C = 0.6G_U$$
$$\tau_I = 0.5\tau_U$$
$$\tau_D = 0.125\tau_U \qquad (7.3)$$

where G_U and τ_U are the ultimate gain and the ultimate period respectively. However, for both first order and second order processes without dead time, equation (7.3) could not be implemented as G_U and τ_U do not exist, due to the fact that the closed-loop response with proportional only control will not oscillate continuously for any value of G_C. Therefore, as starting values, G_C was set equal to $0.6/K_C$, where K_C is the overall (combined) gain of the process (refer to Section 4.4 for the definition of overall (combined) gain). This makes use of the knowledge that the controller gain is normally set inversely proportional to the process gain (at π radians if the process reaches this phase at a particular frequency other than infinity). τ_U was set equal to the theoretical combined time constant T_C of the process found in Section 4.3.1, since the time constants in both directions are different. τ_U will be termed the controller period for the rest of this chapter. (In later sections, the effects of varying the ratio between G_C and $1/K_C$, and τ_U and T_C will be investigated. The ratio between τ_I, τ_D and τ_U will be kept fixed as in equation (7.3). The optimal values for the controller settings in terms of smoothness and speed of response will then be obtained.)

An example is given in Figure 7.2 of the response to a positive step input (setpoint change from 0 to 1) at time $t = 0$ and a negative step input (setpoint change from 1 to -1) at time $t = 30T$. The process has a gain of unity in both directions ($K_C = 1$) and hence G_C was set equal to 0.6. The time constant is T in the positive direction and $5T$ in the negative direction. τ_U was set equal to $1.92T$ as this is the combined theoretical time constant of the process. τ_I and τ_D were set according to equation (7.3). From the plot of the output c, a very slight derivative kick is observed in response to the positive step input. This is due to the slope of the output not being equal to zero when a step input was applied.

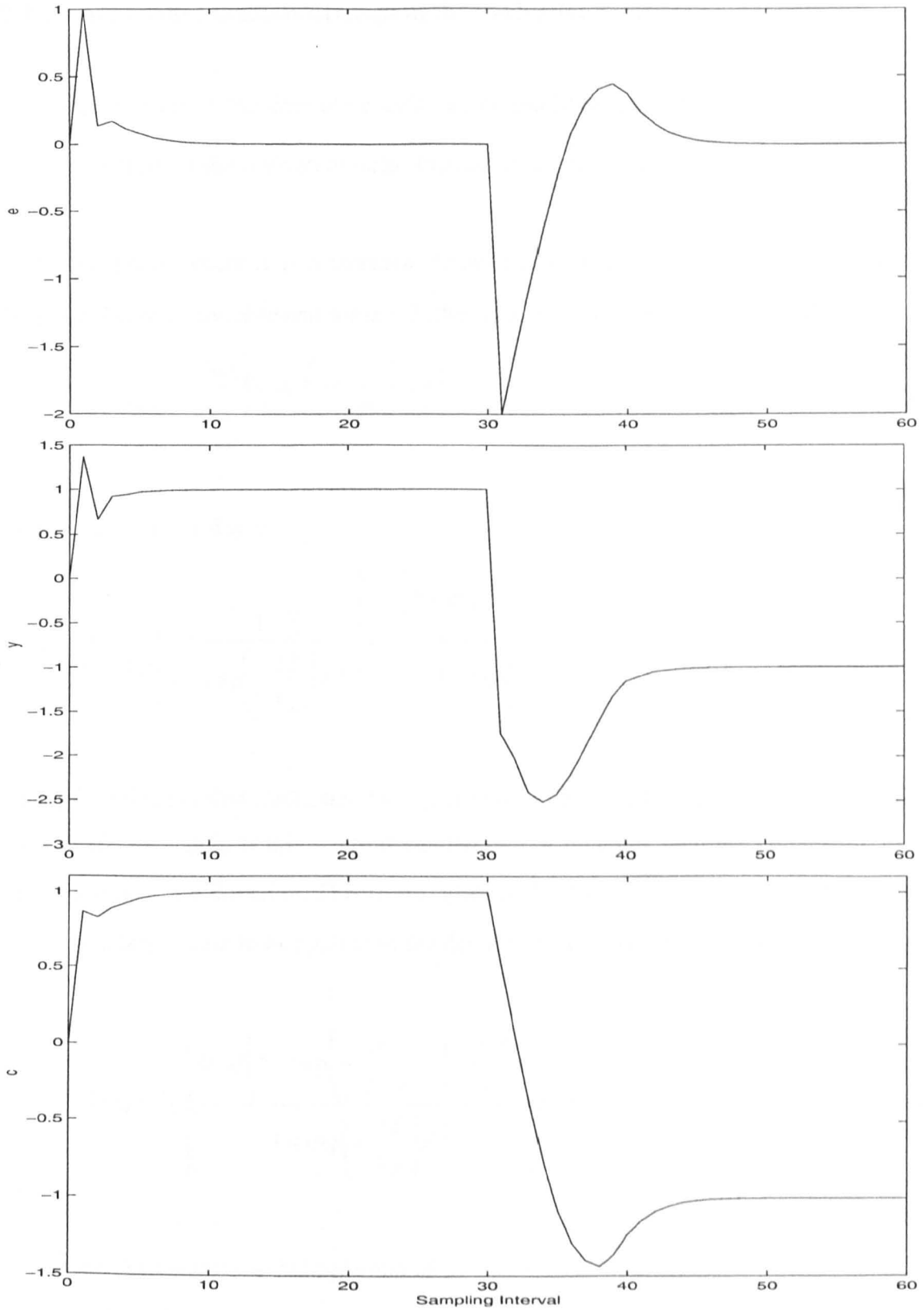

Figure 7.2. Response to a positive step input (setpoint change from 0 to 1) at time $t = 0$ and a negative step input (setpoint change from 1 to -1) at time $t = 30T$ using a direct implementation of the PID. $T_U = T$, $T_D = 5T$ and $\tau_U = 1.92T$.

7.2.1.2 Alternative Implementations of the Derivative Term

The phenomenon of the derivative kick can normally be reduced by using an indirect implementation of the derivative term. Instead of implementing $s\tau_D$, the term $\frac{s\tau_D}{1+\frac{s\tau_D}{\alpha}}$ is implemented, where α is a constant. Choosing α as 3 (since for $\alpha > 3$, the output begins to become unstable and for $\alpha < 3$, the amount of overshoot is increased),

$$\frac{s\tau_D}{1+\frac{s\tau_D}{3}} = \frac{3s\tau_D}{s\tau_D+3} = 3\left(1-\frac{3}{s\tau_D+3}\right) \tag{7.4}$$

The z-transform of this is

$$3-\frac{9}{\tau_D}\left(\frac{1}{1-\exp\left(-\frac{3T}{\tau_D}\right)z^{-1}}\right) = \frac{3\left(1-\exp\left(-\frac{3T}{\tau_D}\right)z^{-1}\right)-\frac{9}{\tau_D}}{1-\exp\left(-\frac{3T}{\tau_D}\right)z^{-1}} \tag{7.5A}$$

The gain, which is first multiplied by G_C, is reduced by a factor of 100 in order that the output will be stable. If this is not done, the output diverges with the application of a step input in either direction. This is also affected by the value of α. A smaller value of α allows a larger gain to be applied to the derivative term. We thus obtain

$$Y(z) = G_C\left(\frac{0.03\left(1-\exp\left(-\frac{3T}{\tau_D}\right)z^{-1}\right)-\frac{0.09}{\tau_D}}{1-\exp\left(-\frac{3T}{\tau_D}\right)z^{-1}}\right)E(z) \tag{7.5B}$$

Hence the derivative term is implemented as

$$y_n = G_C\left(0.03-\frac{0.09}{\tau_D}\right)e_n - 0.03G_C\exp\left(-\frac{3T}{\tau_D}\right)e_{n-1} + \exp\left(-\frac{3T}{\tau_D}\right)y_{n-1} \tag{7.6}$$

Adding the proportional and integral terms, the PID controller is implemented as

$$y_n = \left(1+\exp\left(-\frac{3T}{\tau_D}\right)\right)y_{n-1} + G_C\left[\left(1+\frac{T}{\tau_I}+0.03-\frac{0.09}{\tau_D}\right)e_n - \left(1+0.03\exp\left(-\frac{3T}{\tau_D}\right)\right)e_{n-1}\right] \tag{7.7}$$

The response of the system to the same input and for the same process as in Section 7.2.1.1 is shown in Figure 7.3. From the figure, the effect of derivative kick has been greatly reduced and is no longer noticeable.

Another implementation of the derivative term applies the derivative action to the process output only [84]. This is in order to avoid a large value of the derivative when the setpoint is changed. The term $\tau_D \frac{de}{dt} = \tau_D\left(\frac{dr}{dt} - \frac{dc}{dt}\right)$ is modified to $-\tau_D \frac{dc}{dt}$.

Using this modification, equation (7.2) can be written as

$$y_n = y_{n-1} + G_C\left[\left(1+\frac{T}{\tau_I}\right)e_n - e_{n-1} - \frac{\tau_D}{T}(c_{n-1} - 2c_{n-2} + c_{n-3})\right] \tag{7.8}$$

The response of the system using this implementation is shown in Figure 7.4. The effect of the derivative kick has been eliminated. Unfortunately, the response to a positive step input is less smooth than that in Figure 7.3. However, it is quicker than those shown in Figures 7.2 and 7.3 especially in response to the negative step input. It was decided upon the criteria of smoothness and speed of response that this is the best controller compared to the other two implementations. (Similar criteria are applied when the term 'best' and 'optimal' are used to describe the type of controller or the controller settings throughout the rest of the chapter.)

It should be noted that the decision regarding the best controller was obtained through the simulation of several different processes. However, only one set of simulation results is presented since similar results were obtained for the other processes.

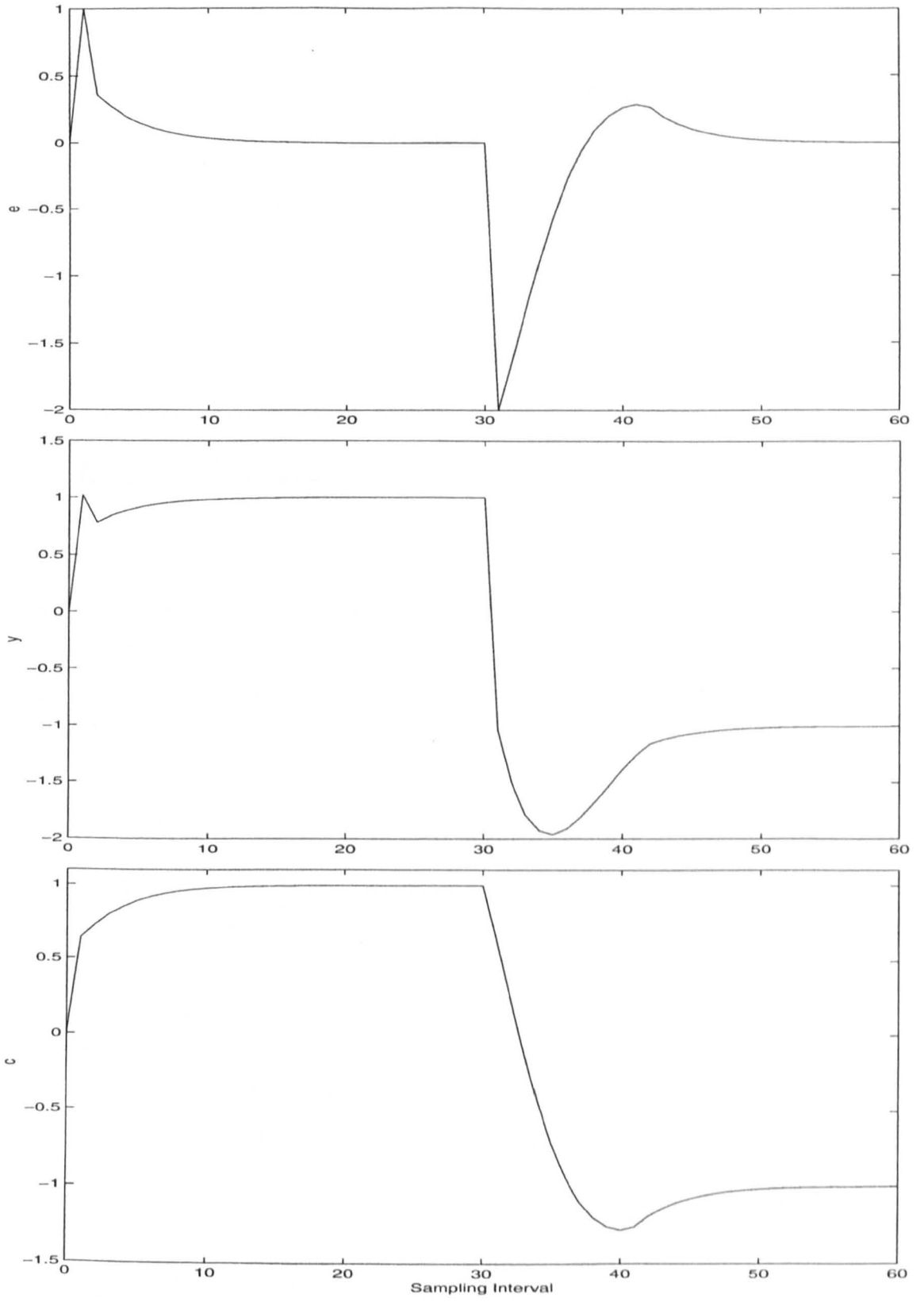

Figure 7.3. Response to a positive step input (setpoint change from 0 to 1) at time $t = 0$ and a negative step input (setpoint change from 1 to -1) at time $t = 30T$ using an alternative implementation of the PID. $T_U = T$, $T_D = 5T$ and $\tau_U = 1.92T$.

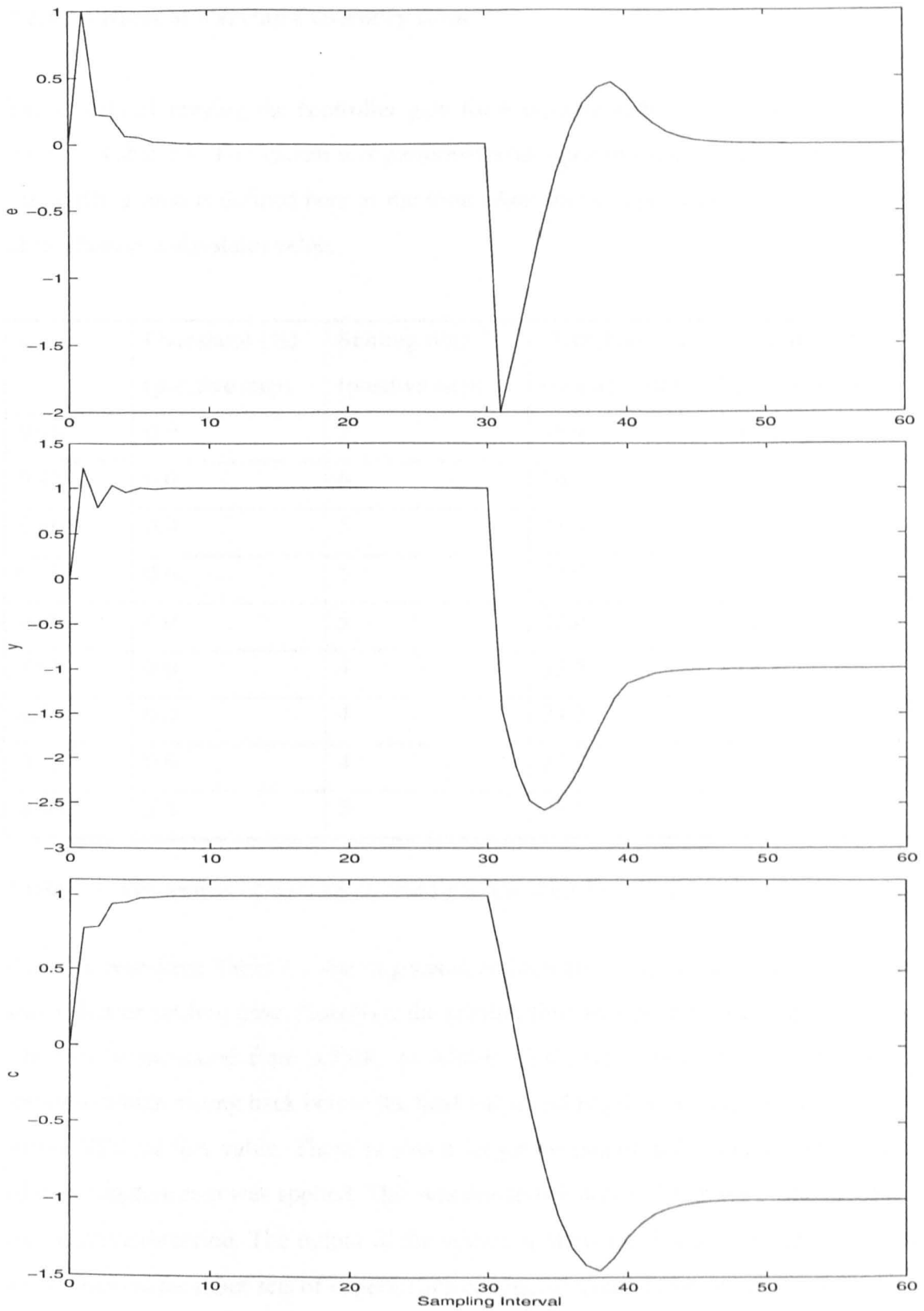

Figure 7.4. Response to a positive step input (setpoint change from 0 to 1) at time $t = 0$ and a negative step input (setpoint change from 1 to -1) at time $t = 30T$. The derivative term is only applied to the process output. $T_U = T$, $T_D = 5T$ and $\tau_U = 1.92T$.

7.2.1.3 Effects of Varying Controller Gain

The effects of varying the controller gain for a process with $T_U = T$ and $T_D = 5T$ are given in Table 7.1. The system was perturbed with a positive and a negative step inputs. The settling time is defined here as the time taken for the output to settle to within 5% of the final (steady-state) value.

$G_C K_C$	Overshoot (%) (positive step)	Settling time / T (positive step)	Overshoot (%) (negative step)	Settling time / T (negative step)
0.40	0.0	7	18.9	15
0.45	0.0	6	20.2	14
0.50	0.0	5	21.4	14
0.55	0.0	5	22.0	13
0.60	0.0	5	23.0	12
0.65	0.0	4	23.5	12
0.70	0.0	4	24.2	11
0.75	0.0	4	24.8	11
0.80	3.4	5	25.0	10

Table 7.1. The effects of varying G_C for a process with $T_U = T$, $T_D = 5T$ and $\tau_U = 1.92T$.

It can be seen from Table 7.1 that in general, an increase in G_C causes a larger overshoot and a shorter settling time. However, the settling time to a positive step input increases when G_C is increased from $0.75/K_C$ to $0.80/K_C$. This was due to the overshoot in the response which swung back below the final value and required a longer time to settle to within 95% of this value. There is also a larger overshoot and a longer settling time when a negative step was applied. This was due to the slower dynamics of the process in the negative direction. The output of the system is shown in Figure 7.5. Similar results were obtained for other sets of experiments conducted using different values of K_C, T_U/T and T_D/T. In general, a good control was achieved by setting G_C to approximately $0.60/K_C$.

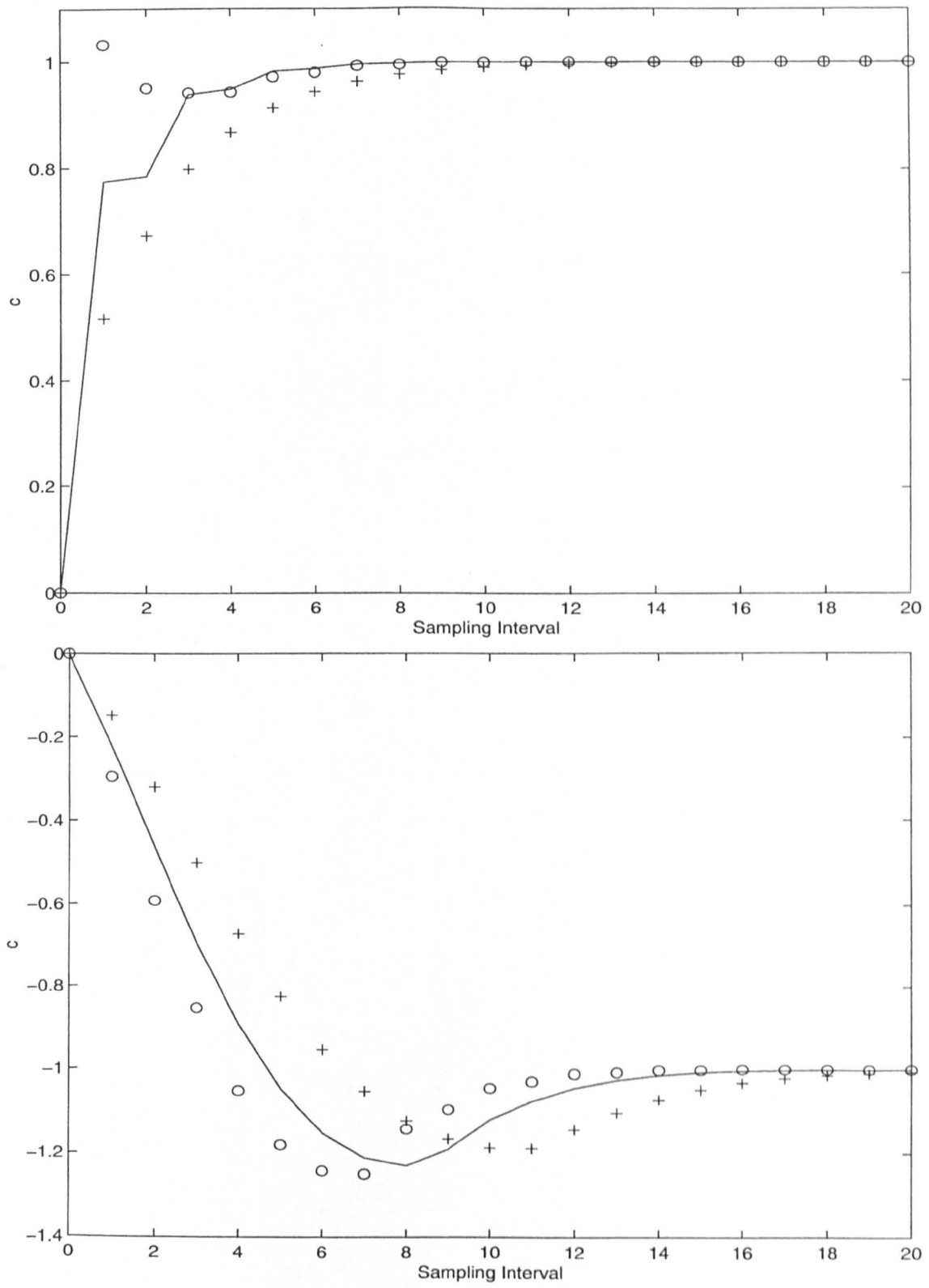

Figure 7.5. The effects of varying G_C for a process with $T_U = T$, $T_D = 5T$ and $\tau_U = 1.92T$.
Solid line: $G_C = 0.60/K_C$; Plusses: $G_C = 0.40/K_C$; Circles: $G_C = 0.80/K_C$.
Top: Response to a positive unit step.
Bottom: Response to a negative unit step.

7.2.1.4 Effects of Varying Controller Period

In the previous sections, τ_U was set equal to the theoretical combined time constant of the process. To investigate the effectiveness of using this value of τ_U for the controller, the performance of the controller with different values of τ_U was evaluated. Results obtained for the same process considered are given in Table 7.2. For this process, T_C is $1.92T$. The system was again perturbed with a positive and a negative step inputs.

τ_U / T	Overshoot (%) (positive step)	Settling time / T (positive step)	Overshoot (%) (negative step)	Settling time / T (negative step)
1.0	13.8	4	36.0	8
1.5	0.0	3	28.4	11
2.0	0.0	5	22.0	13
2.5	0.0	6	17.3	14
3.0	0.0	8	13.9	16
3.5	0.0	10	11.1	17
4.0	0.0	11	8.7	17
4.5	0.0	13	6.8	18
5.0	0.0	15	5.2	18

Table 7.2. The effects of varying τ_U for a process with $T_U = T$, $T_D = 5T$ and $T_C = 1.92T$.

It can be seen from Table 7.2 that in general, a decrease in τ_U causes a larger overshoot and a shorter settling time. There is again a larger overshoot and a longer settling time when a negative step input was applied due to the slower dynamics in the negative direction. The output of the system is shown in Figure 7.6. Similar results were obtained for other sets of experiments conducted using different values of the process parameters, where a better response was achieved with a value of τ_U close to the theoretical combined time constant of the process.

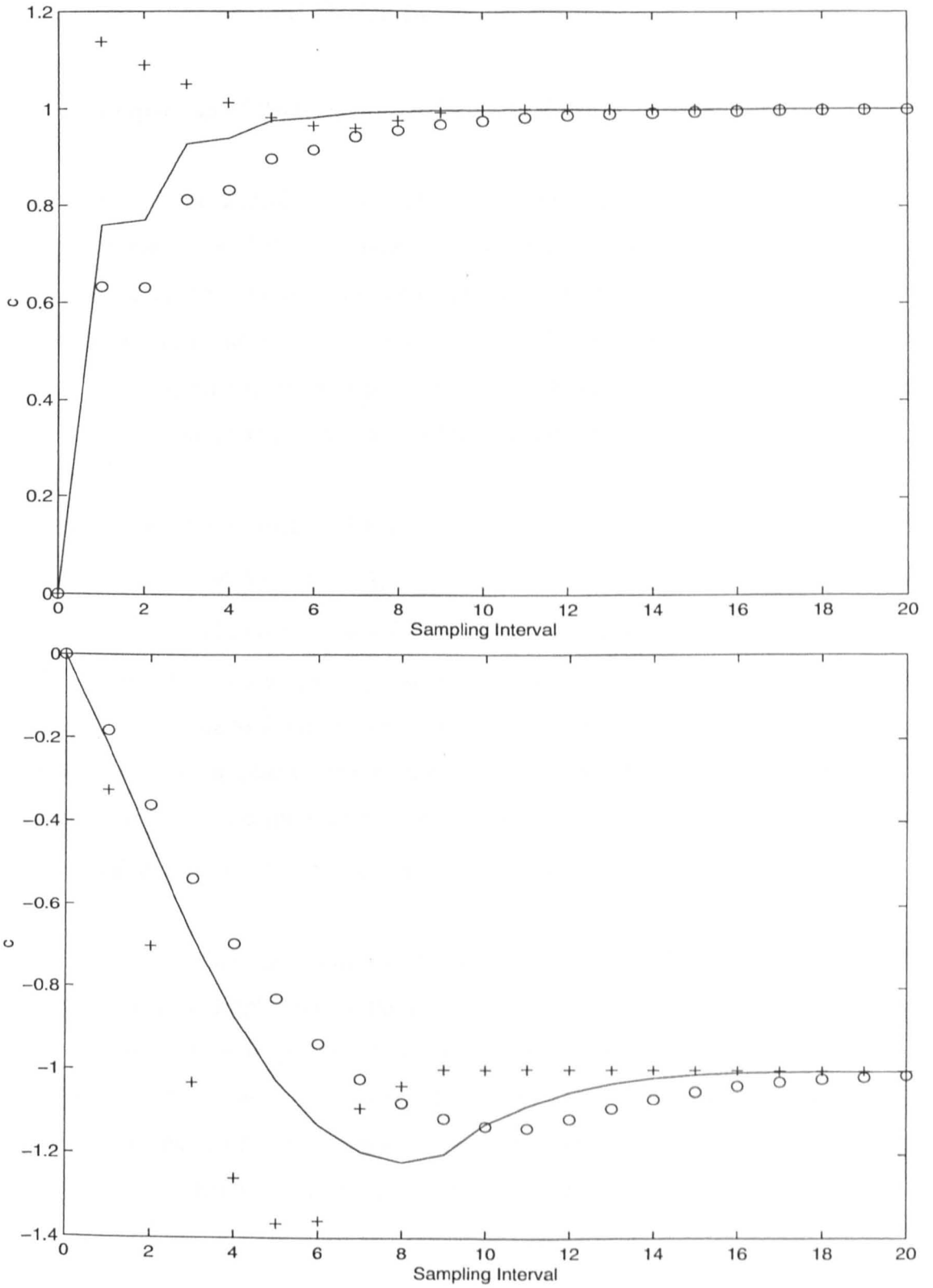

Figure 7.6. The effects of varying τ_U for a process with $T_U = T$, $T_D = 5T$ and $T_C = 1.92T$.
Solid line: $\tau_U = 2.0T$; Plusses: $\tau_U = 1.0T$; Circles: $\tau_U = 3.0T$.
Top: Response to a positive unit step.
Bottom: Response to a negative unit step.

7.2.2 Control of Second Order Process

7.2.2.1 Comparison of Performance of Three Different Controllers

The structure of the control loop is as shown in Figure 7.1. The direct implementation of the PID controller and its two alternative implementations (as discussed in Section 7.2.1.2) were used to control second order processes with direction-dependent dynamics. The performance of each of these were compared. To calculate the output c to the input y of the process, the sampling interval of the path from y to c was again set to twenty times the sampling interval T, as was done in Section 3.4.2.1.

Results obtained for a process with $T_U = T$, T and $T_D = 5T$, $5T$ ($\zeta_U = \zeta_D = 1$, $\omega_{nU} = 1/T$ and $\omega_{nD} = 0.2/T$) and $K_U = K_D = K_C = 1$ in response to a positive and a negative step inputs are shown in Figures 7.7 and 7.8 respectively. (Again, setting $\alpha = 3$ as in Section 7.2.1.2 gave a reasonably fast response with a small overshoot.) G_C was set to $0.3/K_C$ instead of $0.6/K_C$ due to the instability of the system when G_C was set equal to the latter. (The effects of varying the controller gain will be discussed in detail in Section 7.2.2.2.) τ_U was set to $2.62T$ according to the value obtained from ELiS using an inverse-repeat MLB signal as the perturbation signal (see Section 4.3.2).

From Figure 7.7, when using a direct implementation of the PID, the response is seen to be very oscillatory. Implementing the derivative term with an added denominator gives a smoother but slower response. The speed of response is improved with the derivative term applied only to the process output. This response is also reasonably smooth. As in the first order case, this was chosen as the best controller among the three different implementations. Similar conclusions were obtained for other sets of simulations conducted.

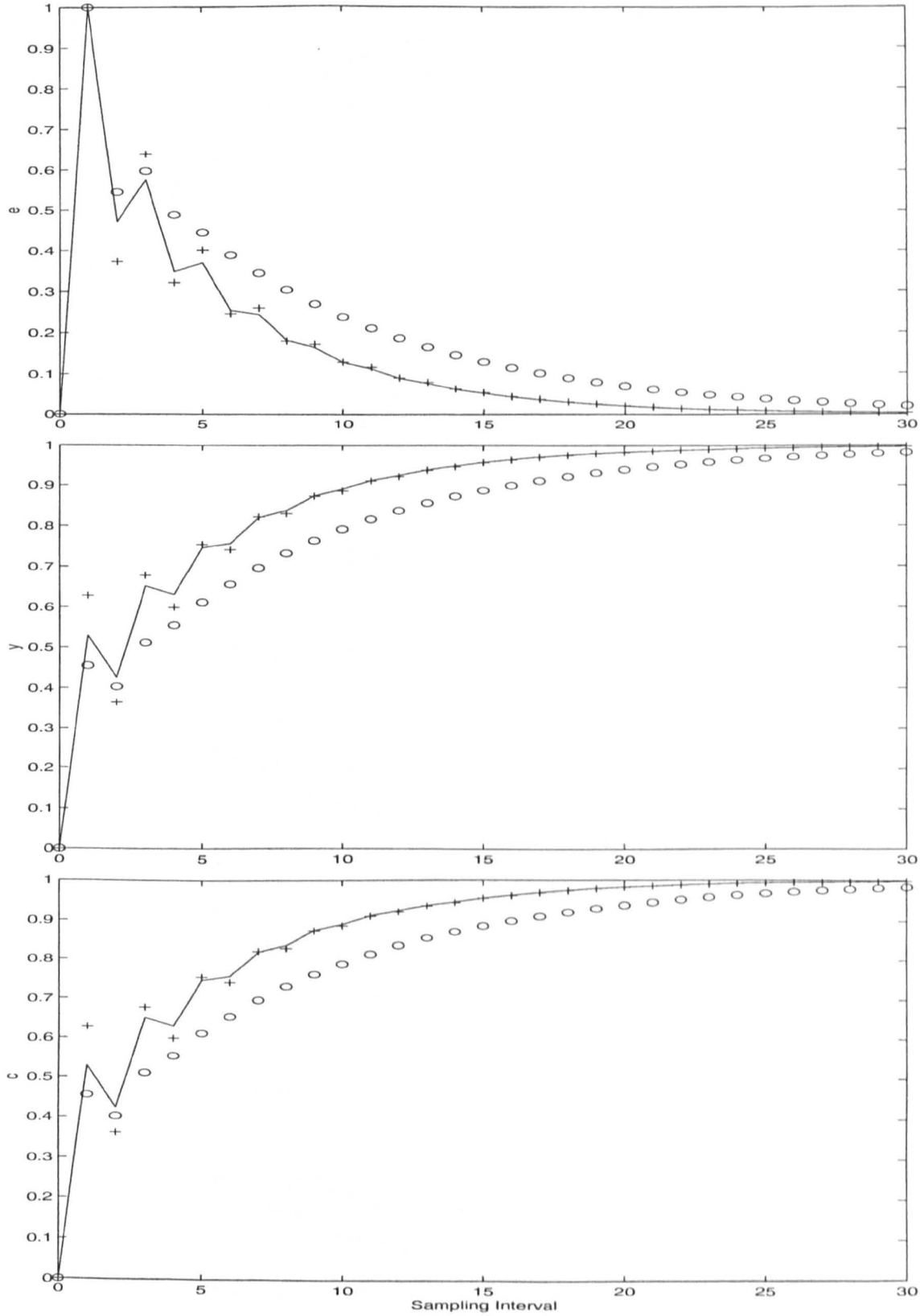

Figure 7.7. Response to a positive unit step input for a process with $T_U = T$, T and $T_D = 5T$, $5T$. Solid line: Derivative term applied only to the process output; Plusses: Direct implementation of PID; Circles: A denominator was added to the derivative term.

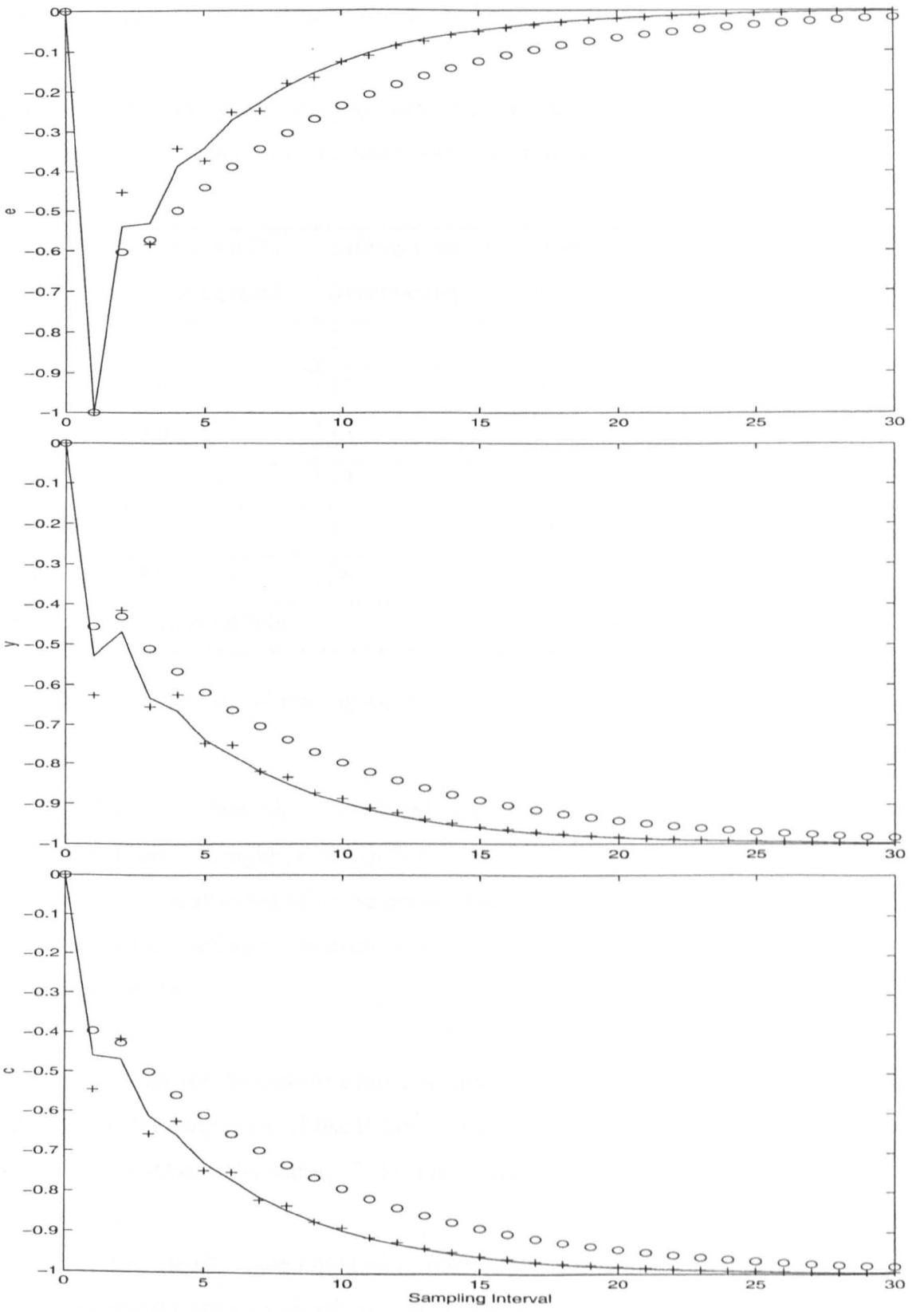

Figure 7.8. Response to a negative unit step input for a process with $T_U = T$, T and $T_D = 5T$, $5T$. Solid line: Derivative term applied only to the process output; Plusses: Direct implementation of PID; Circles: A denominator was added to the derivative term.

7.2.2.2 Effects of Varying Controller Gain

The effects of varying the controller gain for the process considered in 7.2.2.1 are given in Table 7.3. The system was perturbed with a positive and a negative step inputs.

$G_C K_C$	Overshoot (%) (positive step)	Settling time / T (positive step)	Overshoot (%) (negative step)	Settling time / T (negative step)
0.20	0.0	21	0.0	21
0.25	0.0	17	0.0	17
0.30	0.0	15	0.0	14
0.35	0.0	13	0.0	13
0.40	0.2	13	0.0	13
0.45	9.9	24	5.6	21
0.50	system unstable		system unstable	

Table 7.3. The effects of varying G_C for a process with $T_U = T$, T and $T_D = 5T$, $5T$. $\tau_U = 2.62T$.

From Table 7.3, when G_C is increased from $0.20/K_C$ to $0.35/K_C$, the settling time decreases. There is a slight overshoot at $G_C = 0.40/K_C$. When G_C is increased to $0.45/K_C$, there are large oscillations in the response. The settling time increases due to the time required for the oscillations to decrease in amplitude. Further increasing G_C to $0.50/K_C$ leads to instability.

The response for this process to a positive unit step input is shown in Figure 7.9. It can be seen that the amplitude of oscillation increases with an increase in G_C. The optimal response was obtained by setting G_C to approximately $0.25/K_C$.

Almost similar results were obtained in response to a negative step input. The settling time was shorter than or equal to that in the positive direction despite the slower dynamics in the negative direction. This was in fact due to smaller oscillations in the

negative direction as a result of the larger time constants. Similar results were also obtained using several different second order processes.

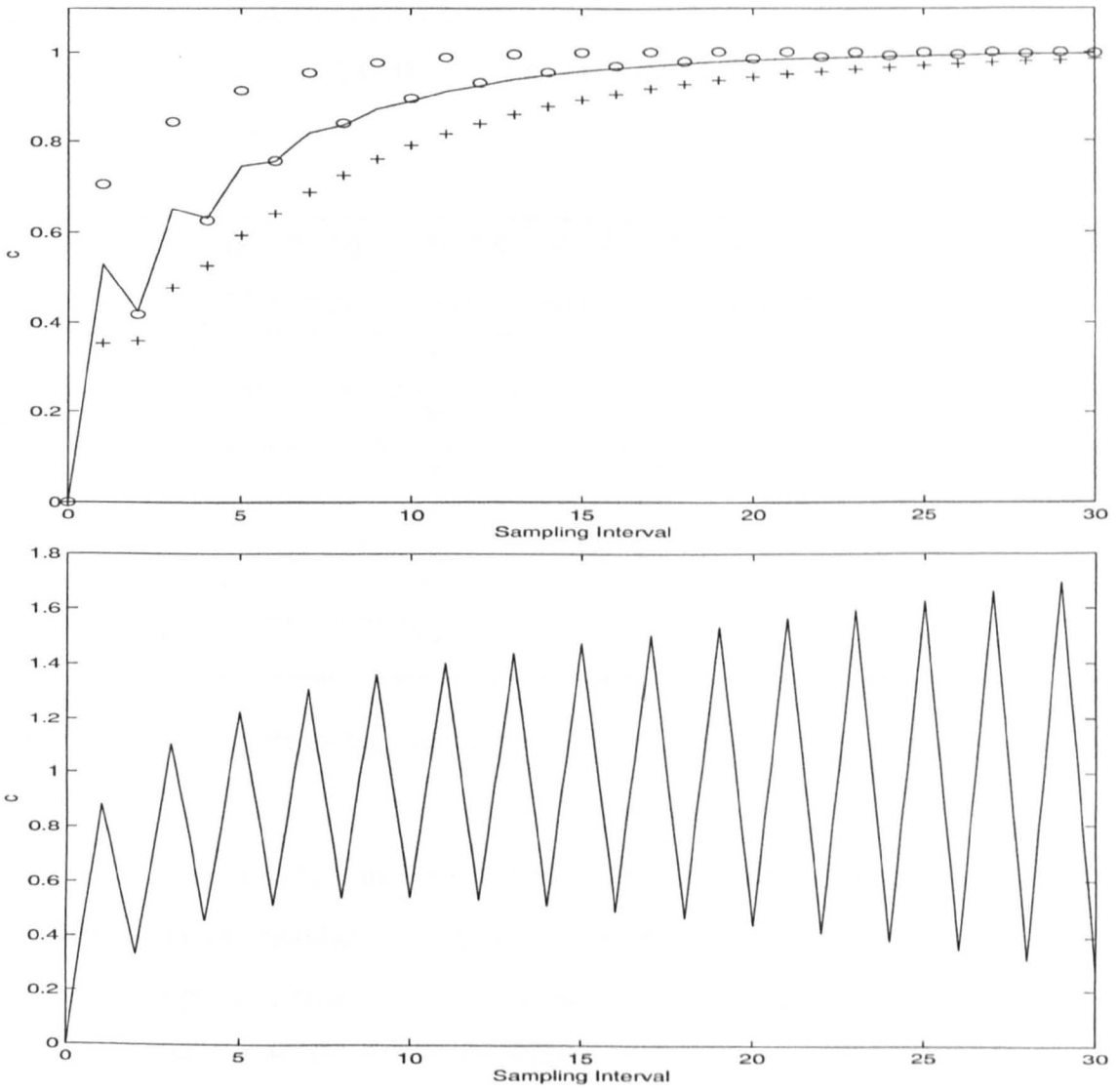

Figure 7.9. Positive unit step response for a process with $T_U = T$, T and $T_D = 5T$, $5T$. $\tau_U = 2.62T$.
Top: Solid line: $G_C = 0.30/K_C$; Plusses: $G_C = 0.20/K_C$; Circles: $G_C = 0.40/K_C$.
Bottom: $G_C = 0.50/K_C$.

7.2.2.3 Effects of Varying Controller Period

The controller period τ_U was varied to find the optimum value for the control of a second order process with direction-dependent dynamics. Results obtained for the same

process considered are given in Table 7.4. The combined time constants found using ELiS are two time constants each of $2.62T$. The controller gain G_C was set to $0.25/K_C$ as this was found to give the optimal response for the controller in the sense that the amplitude of oscillations was reduced without compromising much of the response speed (see Section 7.2.2.2). The system was again perturbed with a positive and a negative step inputs.

τ_U / T	Overshoot (%) (positive step)	Settling time / T (positive step)	Overshoot (%) (negative step)	Settling time / T (negative step)
0.5	25.0	8	8.7	5
1.0	0.0	6	0.0	6
1.5	0.0	9	0.0	9
2.0	0.0	13	0.0	13
2.5	0.0	16	0.0	16
3.0	0.0	20	0.0	20

Table 7.4. The effects of varying τ_U for a process with $T_U = T$, T and $T_D = 5T$, $5T$. $T_C = 2.62T$.

From Table 7.4, when τ_U is increased from $1.0T$ to $3.0T$, the settling time increases. The response becomes smoother. Overshoot is observed at $\tau_U = 0.5T$. A large overshoot in response to a positive step input causes a larger settling time, due to the time required for the oscillations to die down. A negative step input causes a relatively smaller overshoot due to the slower dynamics of the process in the negative direction. The settling time in this direction thus continues to decrease as τ_U is decreased from $1.0T$ to $0.5T$.

The response due to a positive step input is shown in Figure 7.10. The optimal value of τ_U was found to be approximately $1.5T$. This value is a compromise between the smoothness and the speed of response. Using this value for the controller period and varying G_C still gave the 'best' value of G_C at approximately $0.25/K_C$.

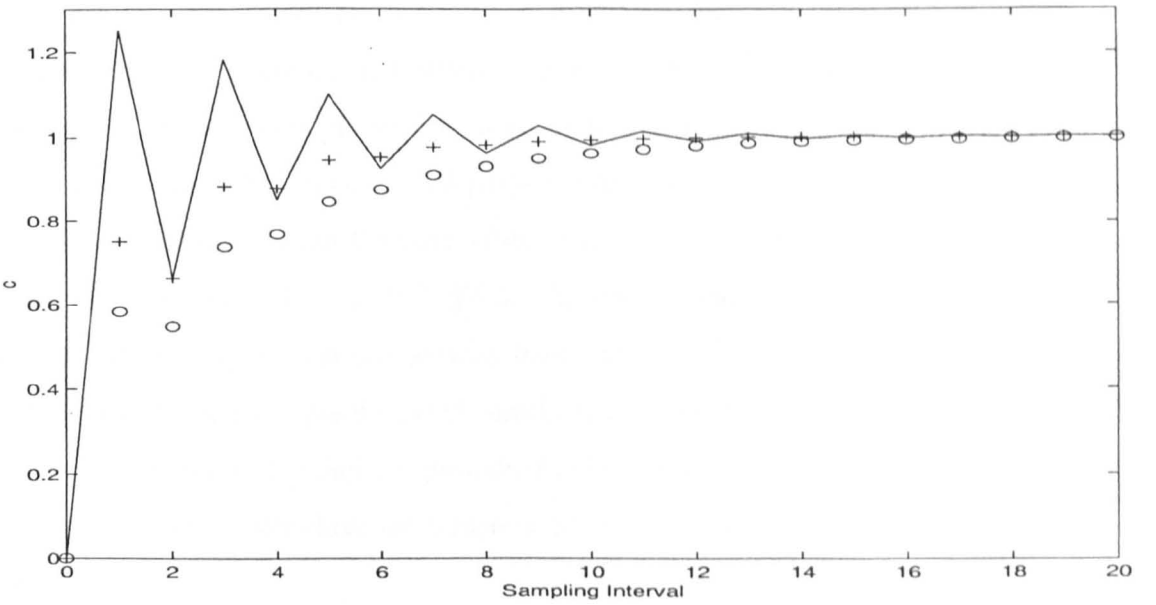

Figure 7.10. Positive unit step response for a process with $T_U = T$, T and $T_D = 5T$, $5T$. $T_C = 2.62T$. Solid line: $\tau_U = 0.5T$; Plusses: $\tau_U = 1.0T$; Circles: $\tau_U = 1.5T$.

7.3 Control of Processes with Direction-dependent Gains

7.3.1 Control of First Order Process

7.3.1.1 Control Using PID

A PID controller with the derivative term applied only to the process output was used to control a process with gain in the positive direction $K_U = 1$, and gain in the negative direction $K_D = 4$. The time constant was set to $5T$ in both directions in order to simplify the problem. The overall gain of the process K_C was found to be 2.5 (see Section 4.4.2) and G_C was set to 0.6/2.5 = 0.24. τ_U was set to $5T$. τ_I and τ_D were chosen according to equation (7.3).

The response to a positive unit step input is given in Figure 7.11. (e, y and c are as defined in Figure 7.1) From the graph, this is clearly unacceptable. There is a positive peak in the output starting at $t = 47$, which reaches a maximum at $t = 50$. From $t = 1$ to $t = 46$, the signal y is larger than c and c tends to increase to y as the gain in the positive

direction is unity. However, at $t = 47$, y is smaller than the value of c at $t = 46$. The output c tries to decrease and when it does so, the gain is now $K_D = 4$. Instead of decreasing, c increases causing a large spike. A negative spike in y ensues and c decreases as a result of this. At this point the gain is still $K_D = 4$ and c attempts to reach $4y$. At $t = 67$, $4y$ is larger than the value of c at $t = 66$. c tends to increase and the gain suddenly changes back to $K_U = 1$. Thus c further decreases and tries to follow y until at $t = 124$ where y again becomes smaller than c at $t = 123$ and the cycle repeats itself. The behaviour of c at other points can be similarly explained. It is interesting to note that the pattern occurring in e, y and c is periodic (and is a form of limit cycles [91 - 93]), except at the beginning where there are transient effects. The period in this case is 77 sampling intervals.

The response to a negative unit step input is given in Figure 7.12. There are spikes in e, y and c as before. From $t = 0$ to $t = 11$, c decreases and the gain is $K_D = 4$. At $t = 12$, $4y$ is larger than c at $t = 11$ and c wants to increase. The gain now becomes $K_U = 1$ and c attempts to follow y. The behaviour of the system is similar to that shown in Figure 7.11. The pattern is again periodic and the period of the limit cycles is 15 sampling intervals, after the transient effects have disappeared.

From Figures 7.11 and 7.12, it can be seen that the spikes in the output are mainly due to the overshoot in c with respect to the value of K_U*y or K_D*y (depending on whether the output is increasing or decreasing) which causes the process gain to change suddenly. The large spikes in y itself lead to a much higher chance of this occurring (a change in the sign of $(K_U*y - c)$ or $(K_D*y - c)$, depending on the direction of the output).

The period of oscillation increases with an increase in the difference between K_U and K_D. This is caused by the larger overshoot as this difference increases. Also, the overshoot is smaller and the period of oscillation is shorter when the step input is applied in the direction with a larger gain. Five sets of results are given in Table 7.5. All these processes has $T_U = T_D = 5T$. The overshoot here is defined as the maximum of the absolute value of the output for unity step inputs.

K_U	K_D	Overshoot (positive step)	Period / T (positive step)	Overshoot (negative step)	Period / T (negative step)
1	2	1.38	56	1.19	14
1	3	1.69	65	1.33	14
1	4	1.98	77	1.45	15
1	5	2.27	104	1.55	16
2	3	1.22	54	1.11	13

Table 7.5. Variation in the overshoot and the period of oscillation for a first order process with $T_U = T_D = 5T$.

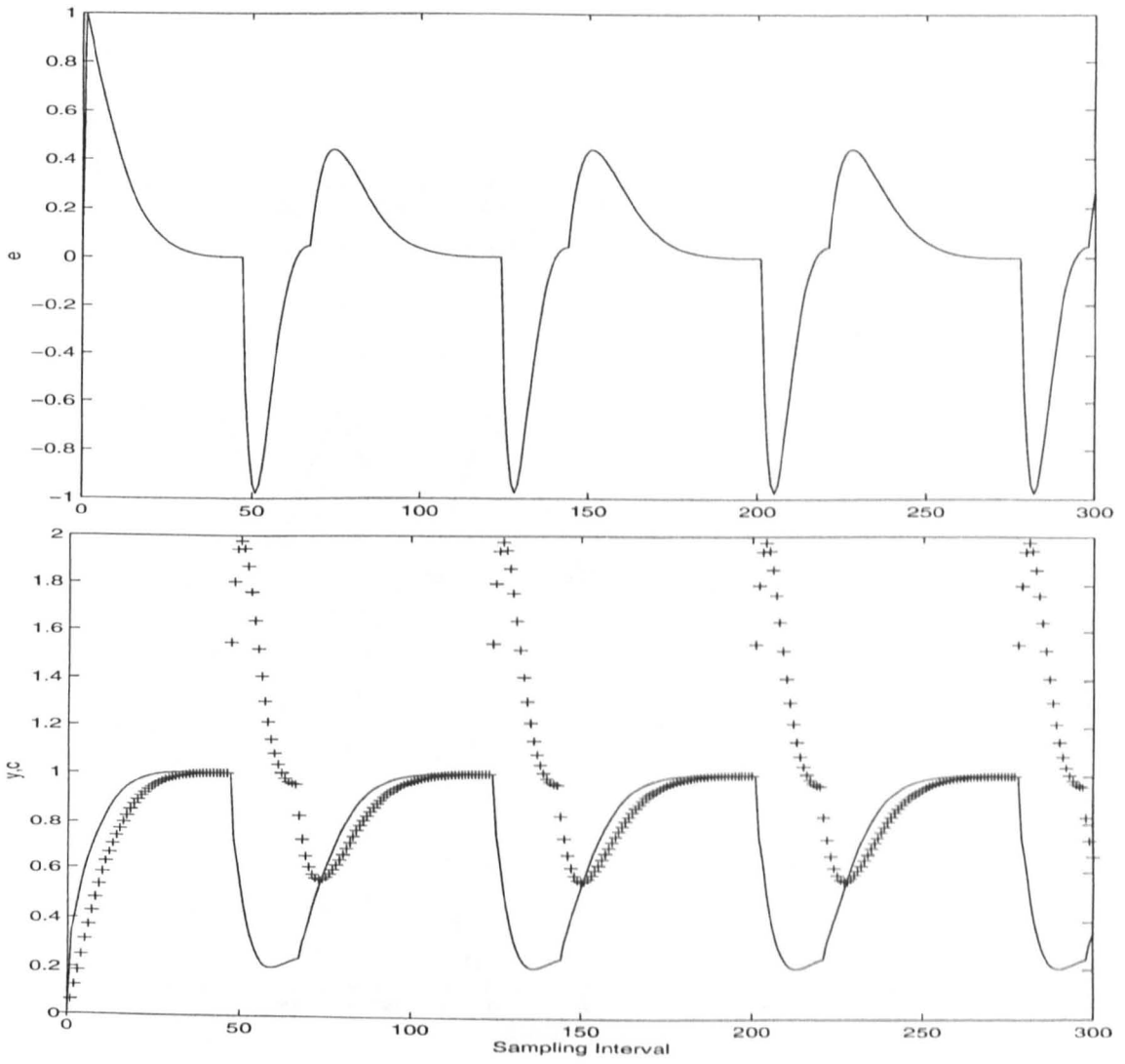

Figure 7.11. Response to a positive unit step input for a process with $K_U = 1$, $K_D = 4$ and $T_U = T_D = 5T$. In the lower figure, plusses represent c and the solid line represents y.

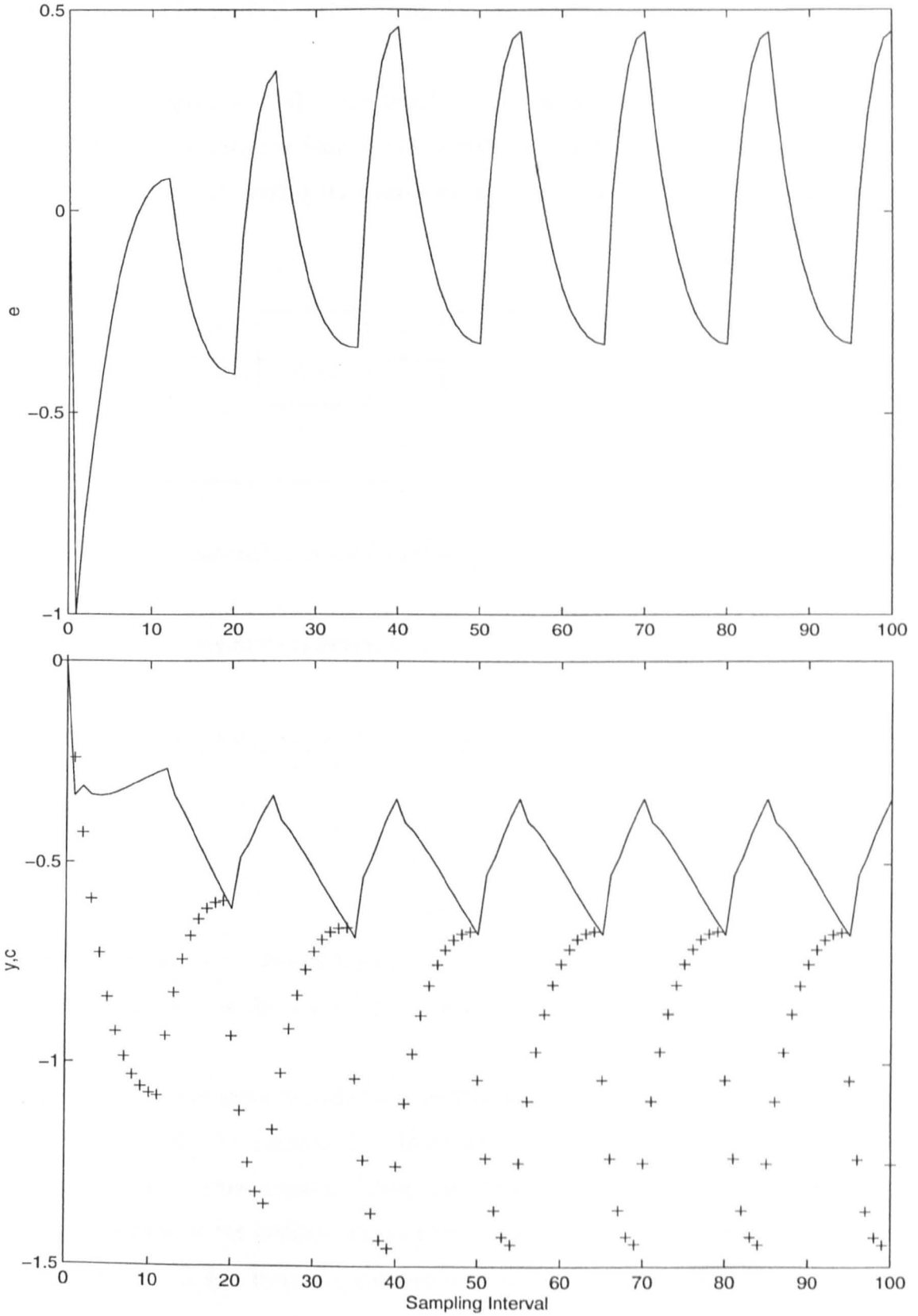

Figure 7.12. Response to a negative unit step input for a process with $K_U = 1$, $K_D = 4$ and $T_U = T_D = 5T$. In the lower figure, plusses represent c and the solid line represents y.

7.3.1.2 Control Using PID with Moving Averager

A moving averager [94 - 96] with a small gain was added in between the PID controller and the process as shown in Figure 7.13 in order to smooth the signal y and scale it to a small value, hence preventing the overshoot in c with respect to the input of the process m.

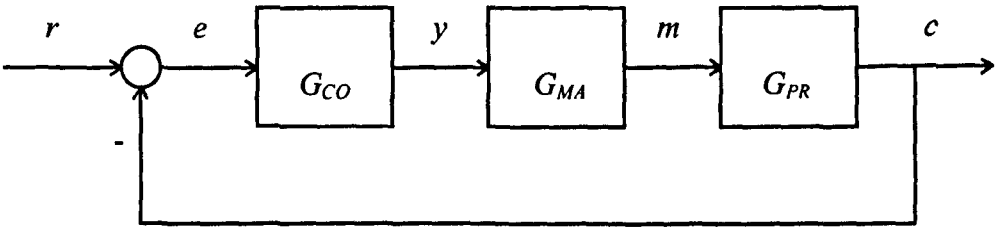

Figure 7.13. Structure of the control loop with a moving averager.

The algorithm for the moving averager is

$$m_n = G_M(y_n + y_{n-1} + y_{n-2} + y_{n-3} + y_{n-4}) \tag{7.9}$$

and

$$G_M = \frac{1}{40}$$

This value of G_M was chosen using trial-and-error. A larger value of G_M will cause instability (periodic oscillation) while a smaller value reduces the speed of response.

The response of the same process to a positive and a negative step inputs are shown in Figures 7.14 and 7.15 respectively. The output is now considerably smoother at the expense of a much slower speed of response. This is due to the very small value for G_M. The settling time in the positive and negative directions are $240T$ and $51T$ respectively. The response is much faster in the negative direction due to the larger gain in this direction.

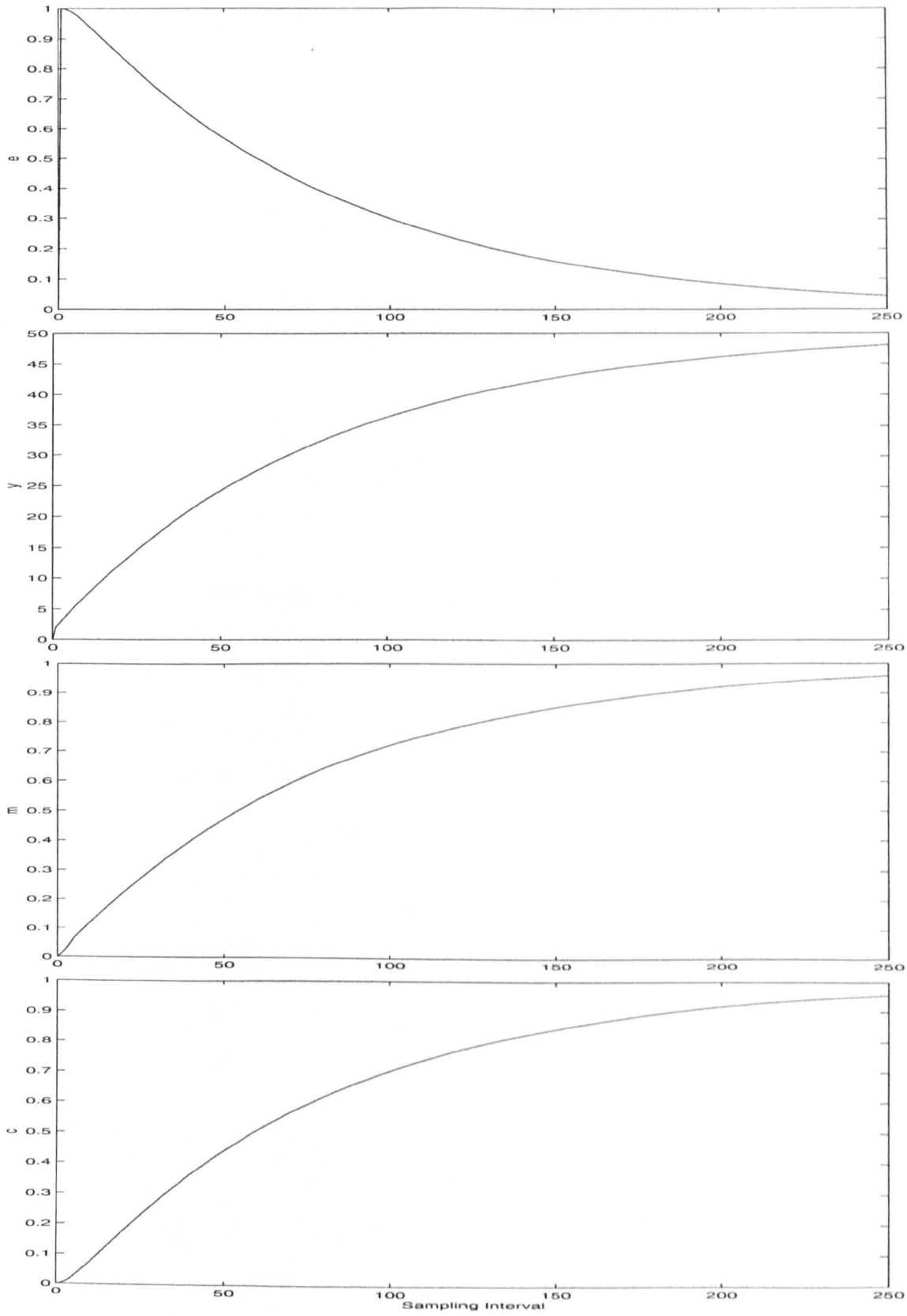

Figure 7.14. Response to a positive unit step for a process with $K_U = 1$, $K_D = 4$ and $T_U = T_D = 5T$. A moving averager had been added.

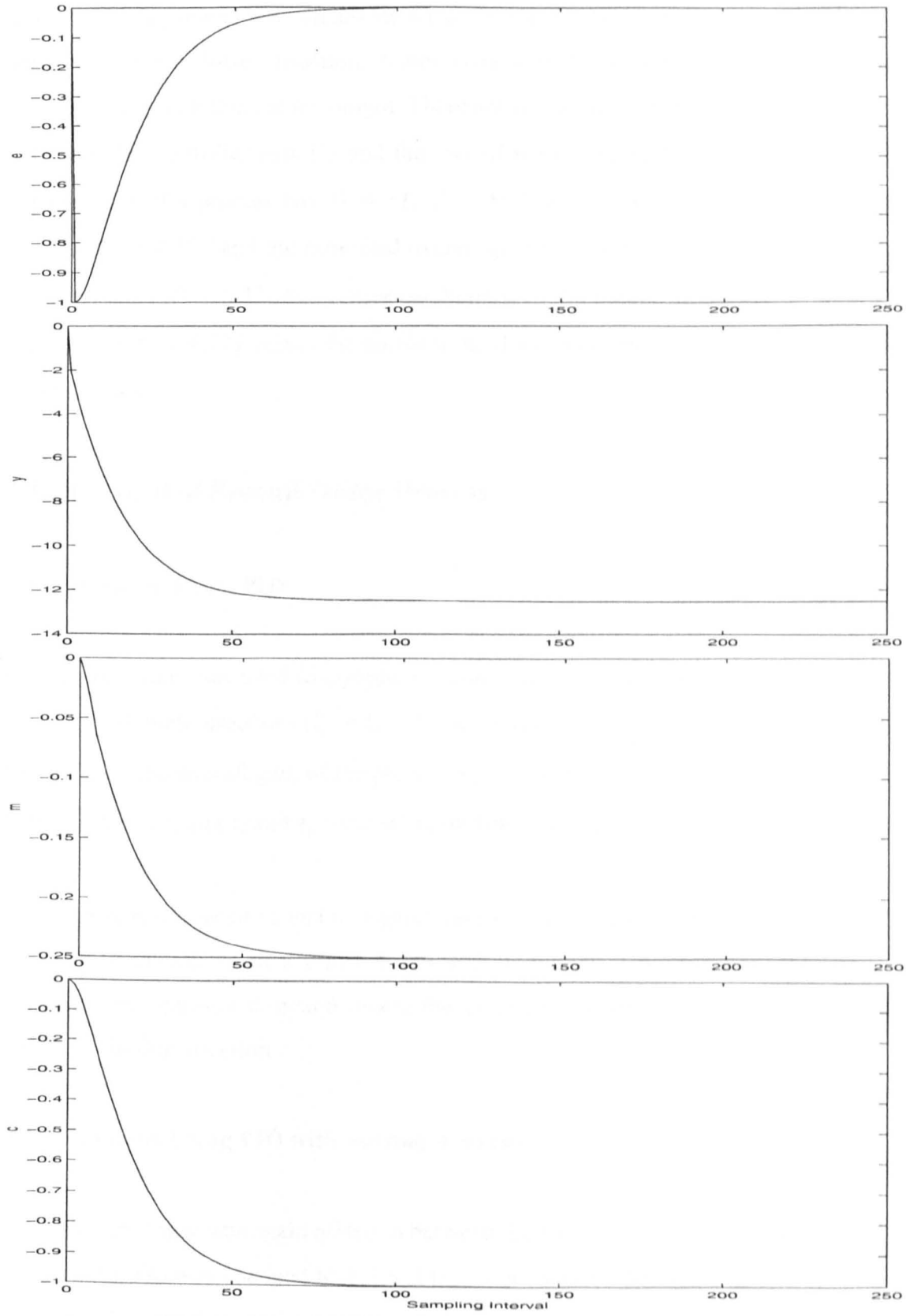

Figure 7.15. Response to a negative unit step for a process with $K_U = 1$, $K_D = 4$ and $T_U = T_D = 5T$. A moving averager had been added.

If the time constants in both directions are different, the worst response occurs when the larger gain is the slower direction. A step change in the direction with the larger gain could lead to oscillations in the output. There are two ways to overcome this. The first is to reduce the controller gain G_C and the second is to increase the controller period τ_U. For example, if a process has $T_U = 3T$, $T_D = 10T$, $K_U = 1$ and $K_D = 4$, the theoretical value of T_C is $4.77T$ and the estimated overall gain K_C is 1.87. With τ_U set to $4.77T$ and G_C set to $0.6/1.87 = 0.32$, this causes oscillations in the output. Reducing G_C to $0.2/K_C$ and keeping $\tau_U = 4.77T$ solves the problem. So does changing τ_U to $11T$ and keeping G_C equal to $0.6/K_C$.

7.3.2 Control of Second Order Process

7.3.2.1 Control Using PID

A PID controller was used to control a second order process with two time constants each of $5T$ in both directions ($\zeta_U = \zeta_D = 1$, $\omega_{nU} = \omega_{nD} = 0.2/T$), and with the gains $K_U = 1$ and $K_D = 4$. The overall gain of the process K_C is 2.5. τ_U was set to $5T$ while G_C was set to $0.25/2.5 = 0.1$, and τ_I and τ_D were set according to (7.3).

The responses to a positive and a negative unit step inputs are given in Figure 7.16. The response is smooth when a positive unit step is applied. Unfortunately, the above is untrue in the opposite direction where the output is extremely oscillatory due to the large gain in this direction.

7.3.2.2 Control Using PID with Moving Averager

A moving averager was again added in between the PID controller and the process as in Figure 7.13. G_M was changed to 1/5 and this was obtained through trial-and-error. The responses to a positive and a negative step inputs are shown in Figures 7.17 and 7.18 respectively. The output is no longer oscillatory and there is no overshoot of c with

respect to m. Thus there is no change in the process gain in response to a step input. A disadvantage of using this control method is the slow response. The settling time is $75T$ in the positive direction and $18T$ in the negative direction.

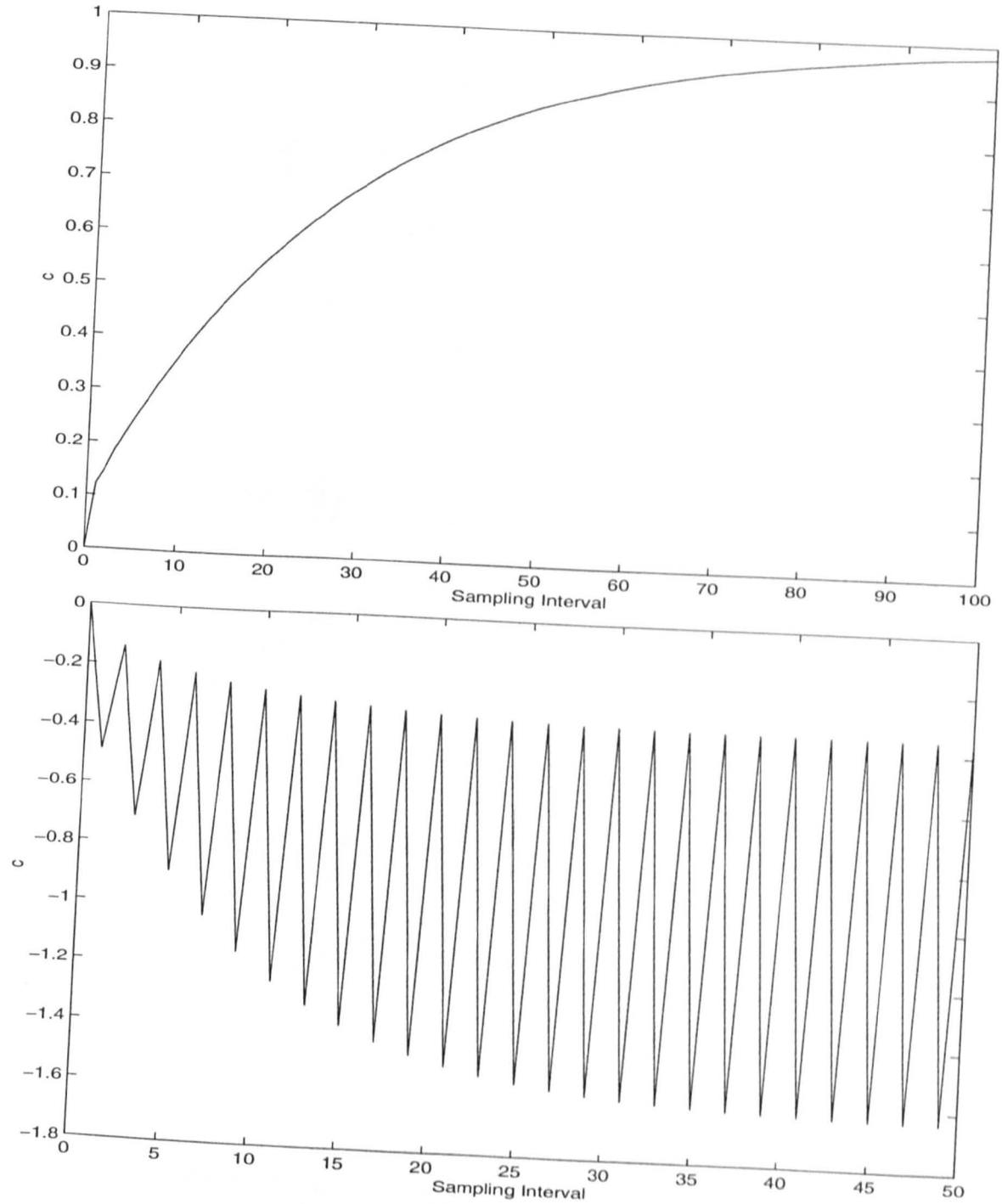

Figure 7.16. Output response of a process with $K_U = 1$, $K_D = 4$ and $T_U = T_D = 5T$, $5T$.
Top: Response to a positive unit step.
Bottom: Response to a negative unit step.

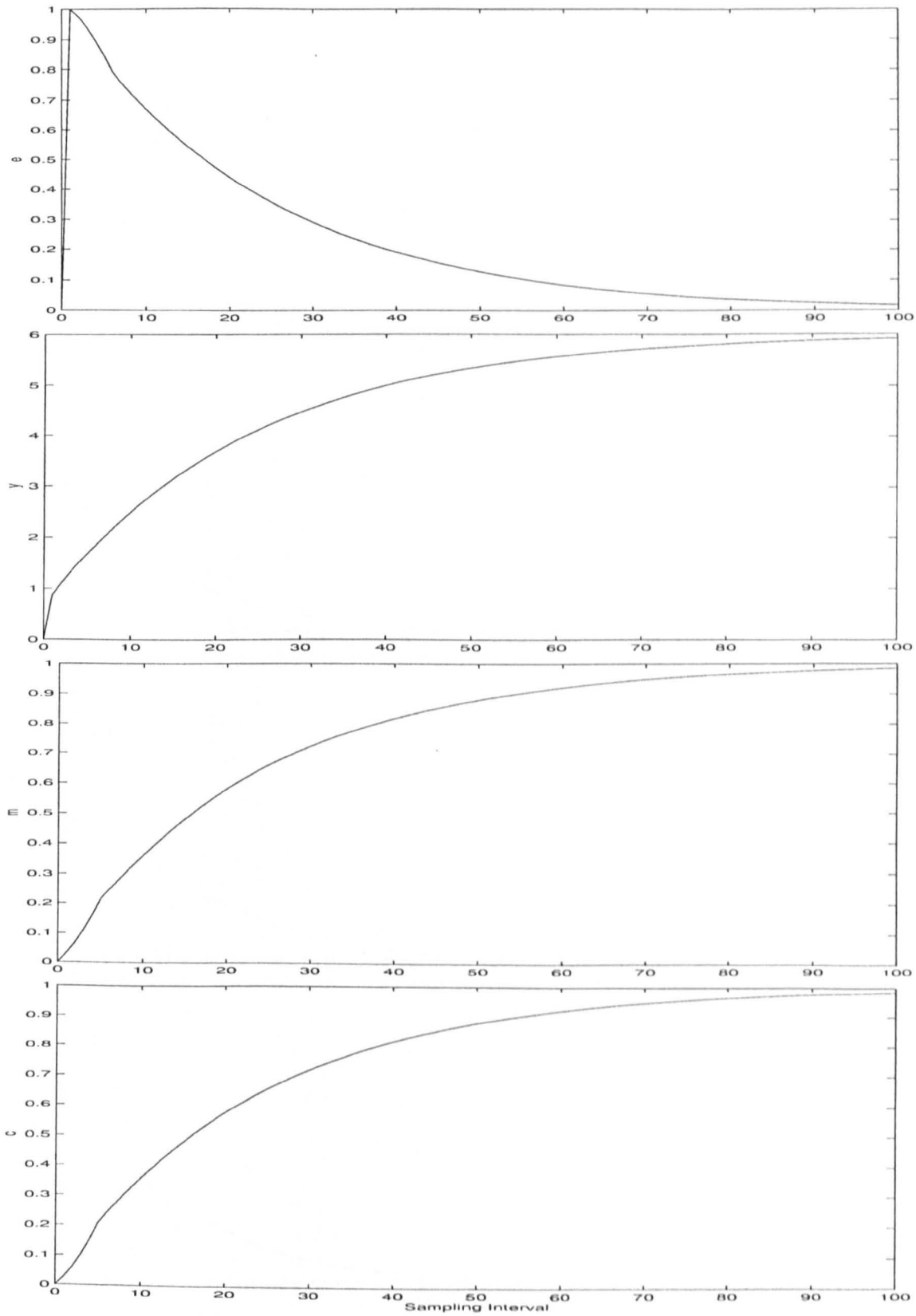

Figure 7.17. Response to a positive unit step for a process with $K_U = 1$, $K_D = 4$ and $T_U = T_D = 5T$, $5T$. A moving averager had been added.

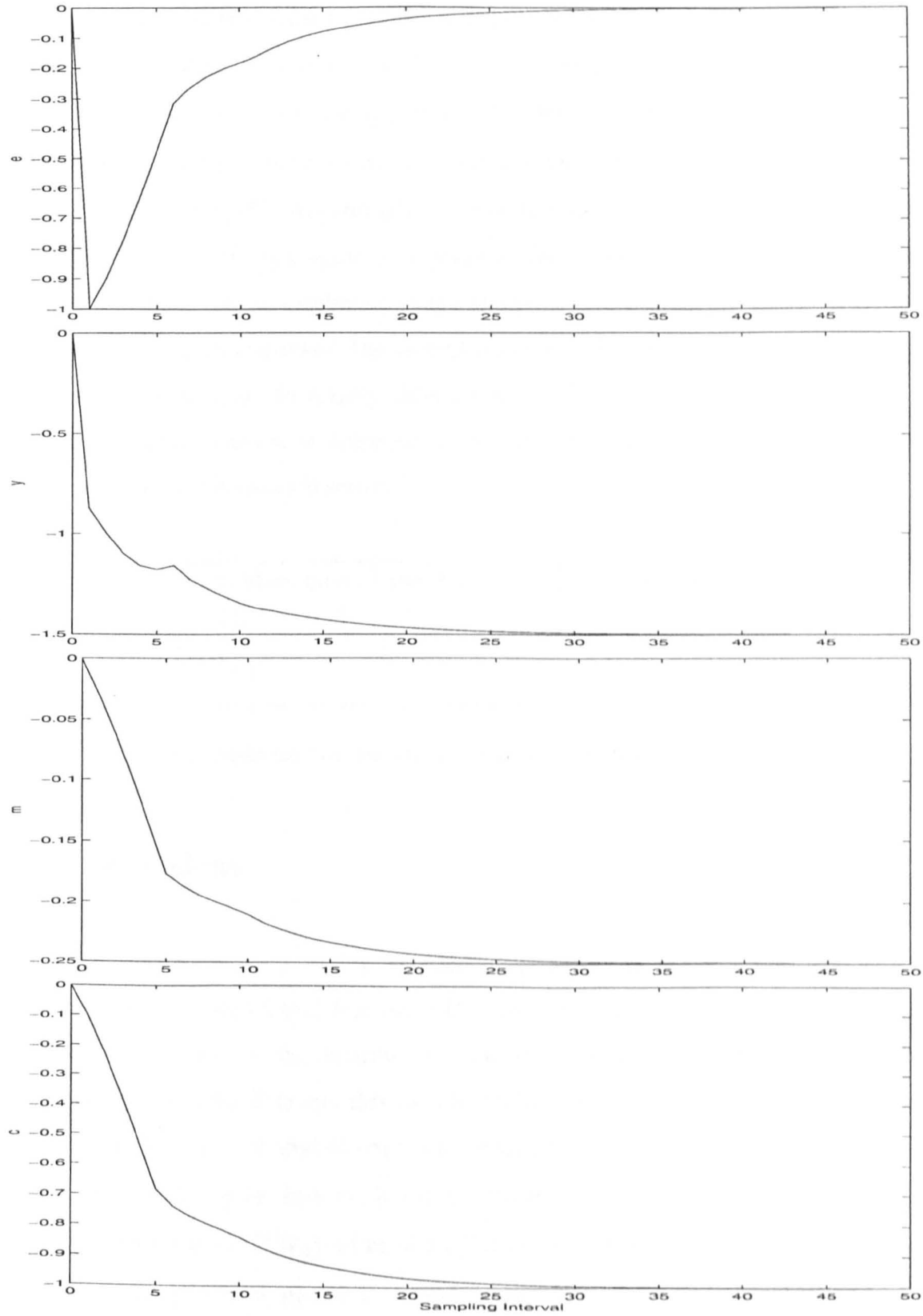

Figure 7.18. Response to a negative unit step for a process with $K_U = 1$, $K_D = 4$ and $T_U = T_D = 5T$, $5T$. A moving averager had been added.

The controller was then used to control a process with different time constants in the positive and negative directions where $T_U = 3T$, $3T$ and $T_D = 10T$, $10T$ (corresponding to $\zeta_U = \zeta_D = 1$, $\omega_{nU} = 0.33/T$ and $\omega_{nD} = 0.1/T$). The gains were $K_U = 1$ and $K_D = 4$. Estimation using ELiS gave T_C as two time constants each of $7.71T$ and an overall process gain K_C of 2.42. Accordingly, τ_U was set equal to $7.71T$ and G_C was set to 0.25/2.42 = 0.10. The responses to a positive and a negative unit step inputs were smooth and there was no oscillation in the response. Exchanging the values of K_U and K_D also gave smooth responses. The settling times are tabulated in Table 7.6. The speed of response is seen to be mainly determined by the process gain in the direction perturbed. Faster response is achieved in the direction with a larger gain even if the dynamics in that direction are slower.

K_U	K_D	Settling time / T (positive step)	Settling time / T (negative step)
1	4	115	29
4	1	30	113

Table 7.6. Settling times for two second order processes with $T_U = 3T$, $3T$ and $T_D = 10T$, $10T$.

7.4 Conclusions

For a first order process with direction-dependent dynamics but with the same gain in both directions, it was found that the PID controller with the derivative term applied only to the output of the process is a better controller compared to the direct implementation of the PID and that with an added denominator in the derivative term. With the derivative term applied only to the output of the process, the speed of response is quicker and derivative kick is eliminated. Setting $G_C = 0.6/K_C$ and setting τ_I and τ_D according to equation (7.3) gives good control in terms of the speed and smoothness of response. The controller period τ_U should be set close to the theoretical value of the combined time constant T_C.

For a second order process with the dynamics, but not the gain, being direction-dependent, the PID controller with the derivative term applied only to the output of the process again gives a reasonably smooth and fast response. The optimum controller gain G_C was found to be equal to approximately $0.25/K_C$. Setting $G_C = 0.6/K_C$ as for the first order process caused instability in most of the cases studied. The controller period can be set to a considerably smaller value than the combined time constant estimated using ELiS but if too small a value is chosen, there will be large oscillations in the output. It should be reiterated here that these are general conclusions drawn from several sets of simulations conducted.

For a first order process with direction-dependent gains, control using PID alone leads to the occurrence of limit cycles. This is chiefly due to changes in the process gain as the output increases and decreases. The problem can be solved by adding a moving averager with a small gain in between the PID controller and the process to smooth the output. The main disadvantage is a slow speed of response. The controller parameters G_C and τ_U can be adjusted if necessary (from the suggested $G_C = 0.6/K_C$ and $\tau_U = T_C$) when the difference between the dynamics in the two directions increases.

For a second order process with the gains different in the two directions, there are oscillations in response to step changes in the setpoint (oscillations may not occur in both directions) if PID control alone is used. Adding a moving averager again provides a solution to the problem. The output is much smoother but the speed of response is now compromised. It was found that the latter is mainly determined by the process gains. The direction with a larger gain has a faster response even if the actual dynamics in that direction are slower.

The overall conclusion is that a moderately good control can be achieved for direction-dependent systems if the process parameters are accurately identified, and the controller parameters are correctly set. However, the above is only possible if such a departure from linearity is first detected and hence recognised in the processes being considered.

Chapter 8

Autotune Control of Processes with Significant Dead Time

8.1 Abstract

An approach to automatic tuning of the parameters of a three-term (PID) controller, involving the addition of a small amplitude signal at the setpoint input of the control loop, has recently been described. The approach processes loop signals in the frequency domain to give an unbiased estimator of the open-loop frequency response of the process, and the controller parameters are then adjusted from frequency response values. Unlike many other autotune methods, it does not require interruption of normal closed-loop operation. In addition, it allows many points on the Nyquist curve to be estimated simultaneously which is useful where the parameters of a process model are to be estimated. In this chapter, the approach is applied to the control of processes with significant dead time. Three different control strategies are examined. In the first, conventional PI control is used (derivative control is ineffective in feedback control of processes with significant dead time), while in the second, a Smith predictor is used for dead time compensation in a loop with PI control. The third control strategy uses a Dahlin algorithm instead of the PI controller. Results from all three control strategies are compared on simulated processes and on a heating process.

8.2 Introduction

Most commercially available three-term (PID) controllers nowadays incorporate an autotune facility in which the proportional gain, and the integral and derivative time constants can be set automatically. In the majority of these, autotuning is done on the basis of a relay test [97 - 99], with the controller parameters being set by rules relating them to the magnitude and frequency of the resulting limit cycle. For a relay without hysteresis, the latter is the frequency at which the process phase is minus 180 degrees. The problem with this method is that it is necessary to switch from the normal closed-loop control to a relay for long enough so that the magnitude and frequency of the limit cycle can be determined.

An alternative approach was proposed in [100] in which, when an update of the controller settings is required, a small amplitude multisine (sum of harmonics) signal is injected at the setpoint. The signals at the controller output and the process output are then processed in the frequency domain to give an unbiased estimate of the open-loop frequency response of the process [101]. The experimental design using this approach was considered further in [102].

The main advantage of this approach is that it is no longer necessary for the normal closed-loop control to be interrupted. A further advantage is that it is possible to obtain the frequency response of the process at frequencies other than that corresponding to minus 180 degrees phase, and this is required by some tuning rules, for example, [103]. The fact that the system response can be obtained at many frequencies in a single multisine test makes this an attractive approach especially when an estimation of the system transfer function is required for purposes of modelling. This technique is robust since it is not dependent on any transient response which is particularly susceptible to the effects of noise. Additionally, it is possible to obtain a reasonably accurate estimate of the steady-state (dc) gain by extrapolation, hence avoiding the need for a separate step response test.

In this chapter, the autotune control of processes with significant dead time is examined. Such processes are quite common in industry and some well known examples are distillation processes [104], coupled-tank systems [105] and the manufacturing of metals, plastics and rubber in sheet form using rolling processes [106]. The process parameters are estimated using least squares [107 - 112] in the frequency domain as this has the advantage that the dead time contributes only to the phase terms in the frequency response [113]. For first order and second order processes, it is shown that the Smith predictor [114, 115] with PI control gives a better performance than that using PI control only. (Derivative action is not included as it becomes ineffective in the control of processes with significant dead time [116 - 118].) The former is also shown to be better than a Dahlin controller [119, 120]. The set of tuning rules proposed can be used for both first order and second order processes, thus simplifying the controller design. In

addition, high order and non-minimum phase processes can be controlled by modelling these as second order processes with dead time. An application to a heating process is used to illustrate the practical nature of the approach.

8.3 Signal Design

In this chapter, the parameters of the processes were estimated in the frequency domain using a multisine signal generated using the command *msinclip* in the Frequency Domain System Identification Toolbox [5]. Following the design strategy suggested in [102], the signal consisted of eight harmonics; its relative phases were optimised for peak-to-peak minimisation in order not to have large amplitude signals superimposed on top of the normal loop signals. This design provided a good compromise between adequate distribution of perturbation frequencies and a relatively short tuning period. The highest harmonic was placed approximately an octave above the critical frequency of the process. (The approximate value of the critical frequency is assumed to be known. Otherwise, a separate relay test will have to be conducted before the test signal is designed.) The number of samples N in the signal was selected to ensure that the sampling frequency was at least 20 Hz; this is the minimum sampling frequency acceptable for discrete control of any of the processes considered in this chapter. There was a minimum of 240 samples, as in [100]. 10000 iterations were carried out in the design of the signal.

When autotuning was required, one and a quarter periods of the multiharmonic signal were added to the reference r. As in [100], the first quarter period was not used in the identification due to the presence of transient effects. From experience, this is sufficiently long for the transient effects to die away. An unbiased estimate of the open-loop frequency response $H(j\omega)$ of the process is given in [101] as

$$\hat{H}(j\omega) = \frac{\hat{S}_{yr}(j\omega)}{\hat{S}_{ur}(j\omega)} \tag{8.1}$$

where u is the input to the process, y is the output, and $\hat{S}_{ur}(j\omega)$ and $\hat{S}_{yr}(j\omega)$ are the estimates of the cross-power-spectral density between r and u, and r and y respectively. With periodic excitations, the estimate $\hat{S}_{yr}(j\omega)$ is given by

$$\hat{S}_{yr}(j\omega) = \frac{1}{p}\sum_{p} R_p^*(j\omega).Y_p(j\omega) \tag{8.2}$$

with $R_p(j\omega)$ and $Y_p(j\omega)$ representing the Fourier coefficients of the pth period of the excitation and closed-loop response respectively. ($*$ denotes complex conjugate.) The estimate $\hat{S}_{ur}(j\omega)$ is similarly defined.

8.4 Smith Predictor

The basic structure of a control loop incorporating the Smith predictor is shown in Figure 8.1. (There are also some modified or extended structures such as those suggested in [107, 121 -124].) In this figure, C(s) is the controller transfer function; G(s) is the process transfer function excluding the dead time L; d is the disturbance (load); and n is the noise. r, u, and y have the same meaning as in Section 8.3 above. The process transfer function including the dead time will be denoted by H(s) and following the notation used in [125], the actual parameters of the transfer function are denoted with a bar on top of the variable.

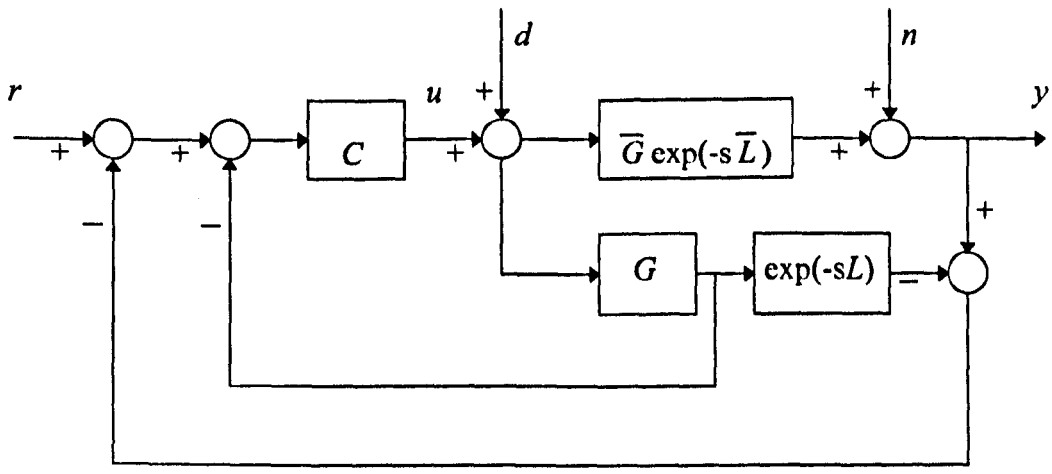

Figure 8.1. Control loop incorporating the Smith predictor.

8.5 Control of a First Order Process

8.5.1 PI Control

Consider a first order plus dead time process with transfer function of the form

$$H(s) = \frac{K_P}{sT_P + 1} e^{-sL} \tag{8.3}$$

where K_P is the steady-state gain and T_P is the time constant. K_P, T_P and L can be calculated using least squares from

$$K_P^2 - |H|^2 \omega^2 T_P^2 = |H|^2 \tag{8.4}$$

$$-\omega L - \tan^{-1} \omega T_P = \angle H \tag{8.5}$$

Many tuning rules have been suggested by different authors to control a first order process. In [125], it was proposed that, assuming that the model of the actual process is accurate, the controller is designed such that the closed-loop response, compensated for dead time, is of the form

$$\frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{1}{sT_a + 1} \tag{8.6}$$

T_a is a user specified parameter and is equal to αT_P; a suitable value for α is between 0.2 to 1 [125]. This gives the parameters of the PI controller as

$$K_C = T_P / K_P T_a = 1/\alpha K_P \tag{8.7A}$$

$$T_I = T_P \tag{8.7B}$$

The tuning rules given by (8.7) give good control of the process, but the design is not robust to noise and inaccurate initial values. It was pointed out in [105, 126] that dead time controllers are mostly sensitive to the mismatches in the dead time. Hence, it was proposed in [107] that the controller gain should be set inversely proportional to the dead time. Incorporating the above ideas, the PI controller used in this chapter was tuned such that

$$K_C = 3/K_P L\beta \tag{8.8A}$$

$$T_I = T_P \tag{8.8B}$$

where β is between 1 and 20. The reason for including a factor of three in the numerator of (8.8A) is so that for most processes, good control is achieved with β set to unity. However, if the process is very noisy or if the actual process and its model are not of the same type or order, such as due to certain cases of reduced order modelling, β should be set >1.

Increasing K_C increases the setpoint overshoot while decreasing K_C increases the amplitude of the response due to a load disturbance. Decreasing T_I causes the response to be faster but it also decreases the robustness of the controller due to a larger integral term. A derivative term was not added as its effect was found to be negligible [118]. Hence, a two term (PI) controller was used with an added advantage of simplicity.

8.5.2 The Dahlin Controller

The Dahlin controller [119] is a discrete algorithm designed specifically for a first order process with dead time. For such a process, the closed-loop step response is also first order with dead time, with a single tuning parameter Q being used to set the closed-loop time constant. The controller algorithm is of the form

$$C(z^{-1}) = \frac{Q}{K_P M} \cdot \frac{1-(1-M)z^{-1}}{1-(1-Q)z^{-1}-Qz^{-(N_R+1)}} \tag{8.9}$$

where $M = 1 - \exp(-T/T_P)$, T is the sampling interval, $Q = 1 - \exp(-T/T_{CL})$, where T_{CL} is the closed-loop time constant, and N_R is the ratio of L/T rounded to the nearest integer. The algorithm contains integral action, since the denominator can be factorised with $1-(1-Q)z^{-1}-Qz^{-(N_R+1)} = (1-z^{-1})(1+Qz^{-1}+Qz^{-2}+\ldots+Qz^{-N_R})$.

8.5.3 Simulation Results

8.5.3.1 Parameter Estimation

As an example, a simulation was carried out for a process with $K_P = 1$, $T_P = 1$s and $L = 10$s (Process J). For this process, the critical gain $K_U = 1.04$ and the critical period $T_U = 22.0$s. An eight-harmonic multisine signal with $N = 2048$ was designed and the sampling frequency was set at 25 Hz. (The signal is shown in Figure 8.2.) This placed the highest harmonic at 0.0977 Hz, which is reasonably close to twice the critical frequency of 0.0456 Hz.

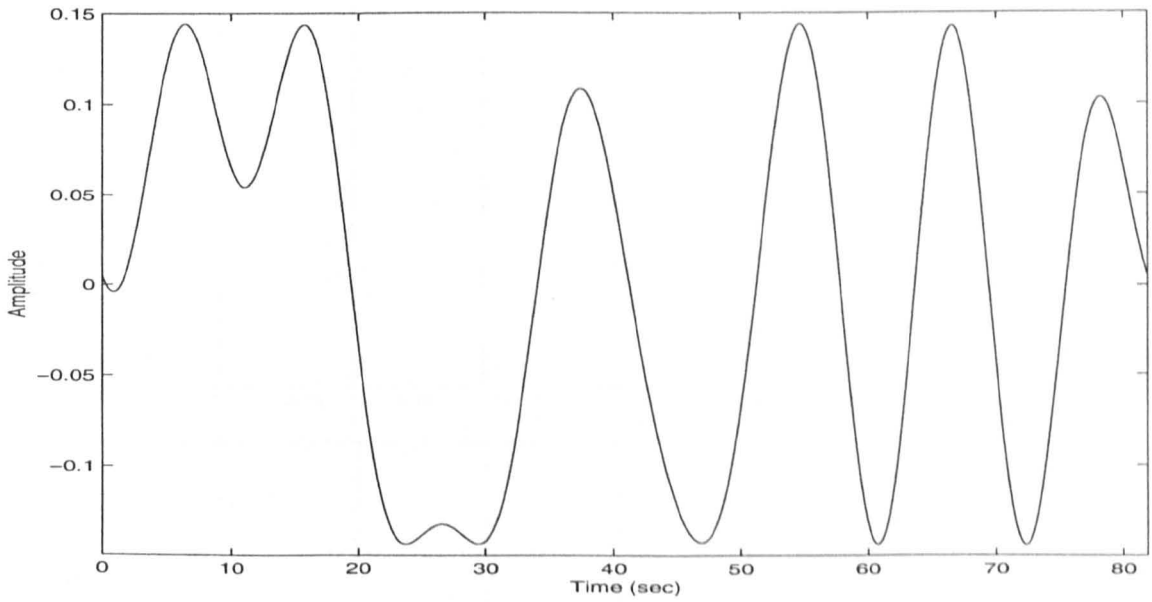

Figure 8.2. Multisine signal with eight harmonics.

The resulting estimates of the process parameters K_P, T_P, L, and the PI controller parameters K_C and T_I (equations (8.8)) are given in Table 8.1 for the cases with no noise added, and a Gaussian noise with signal-to-noise ratios (ratios of r to n in Figure 8.1) of 10dB and 15dB added. It can be seen that the parameter estimates are in close agreement with the theoretical results. The simulations were carried out using SIMULINK in MATLAB [127]. The frequency response identified using the multisine signal are shown in Figure 8.3, for the case with an SNR of 10dB.

	K_P	T_P(s)	L (s)	K_C	T_I(s)
Theoretical	1.00	1.00	10.00	0.30	1.00
No noise	1.00	1.01	9.99	0.30	1.01
SNR of 15dB	1.00	1.01	9.99	0.30	1.01
SNR of 10dB	1.10	1.08	9.92	0.27	1.08

Table 8.1. Process J : Actual and estimated values for process and controller parameters using Smith predictor with PI control. $\beta = 1$.

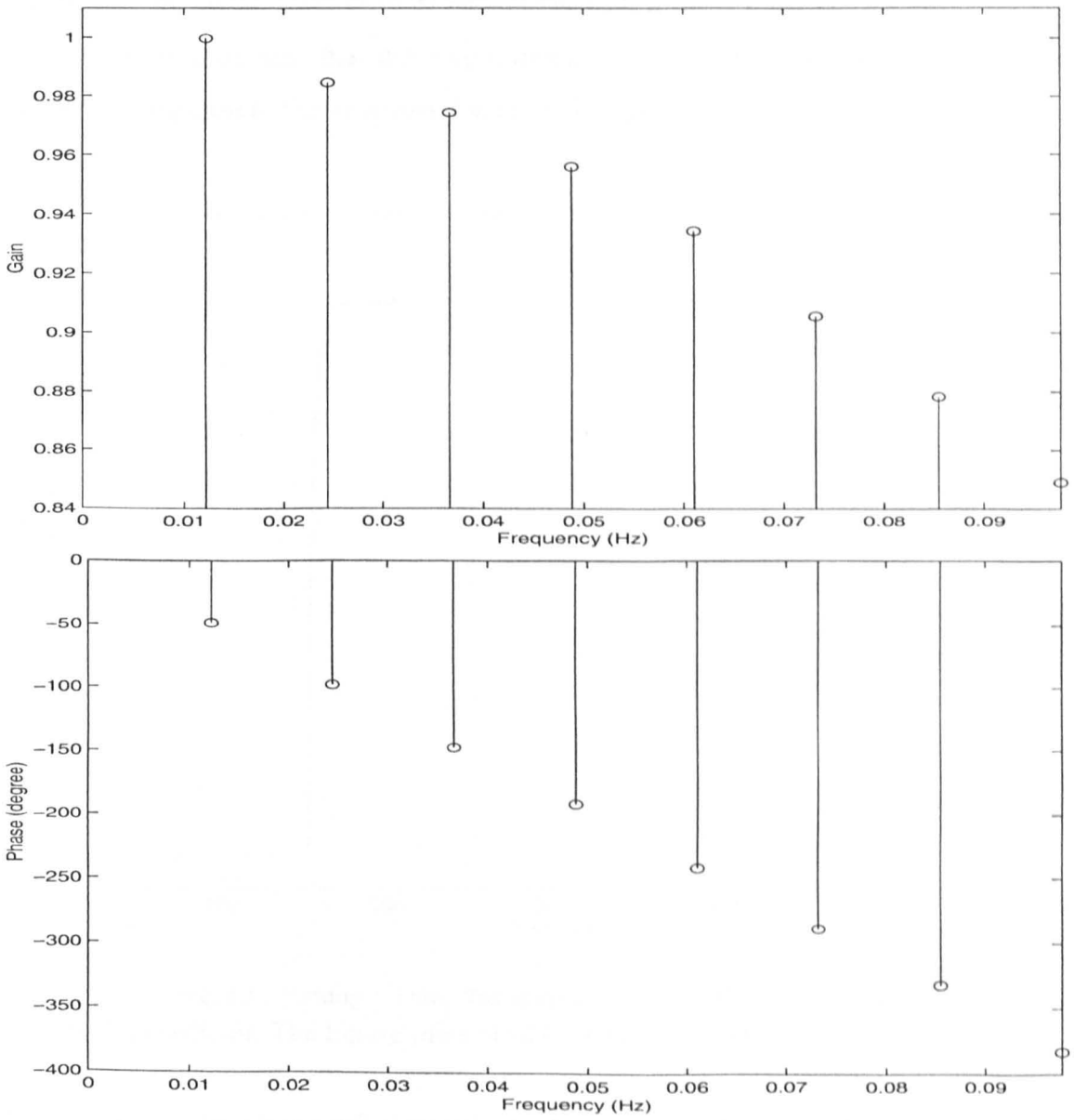

Figure 8.3. Gain response (top) and phase response (bottom) estimates for Process J with an SNR of 10dB.

8.5.3.2 Controller Responses

In Figures 8.4 to 8.7, the responses using different controllers with no noise added are shown. In each case, the tuning phase is carried out between $t = 0$ and $t = 102$s, and the newly tuned controller is then commissioned. A setpoint change of amplitude 5 is introduced at $t = 142$s and a load disturbance of amplitude 1 is introduced at $t = 442$s.

Both Figures 8.4 and 8.5 show responses using a Smith predictor with PI control. In Figure 8.4, the tuning rules of (8.8) were used, with β set to unity (giving $K_C = 0.3$ and $T_I = 1$s). It may be seen that the output quickly settles out to the correct steady-state value. The response to the load disturbance is also good.

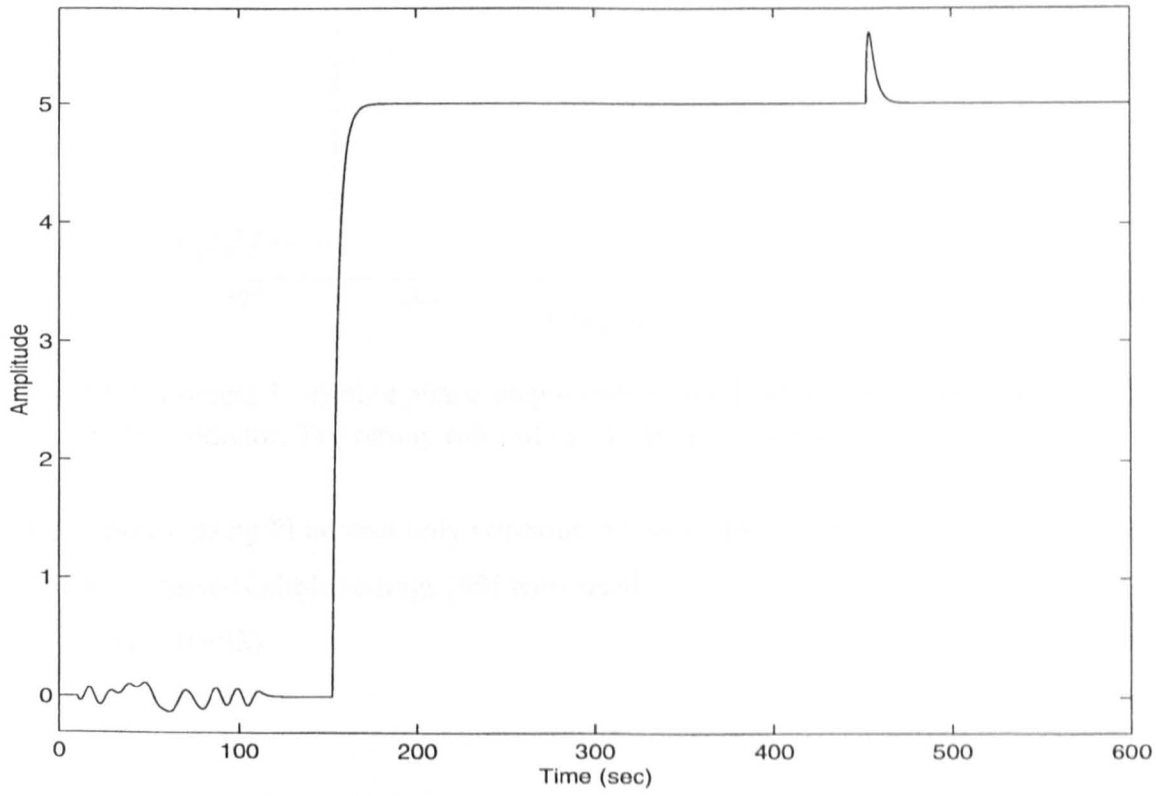

Figure 8.4. Process J : Tuning phase, step response and load response using PI controller with Smith predictor. The tuning rules of (8.8) with $\beta = 1$ were used.

In Figure 8.5, the tuning rules of (8.7) with $\alpha = 1$ were used instead. ($K_C = 1$ and $T_I = 1$s.) The response is seen to be excellent. However, the controller is not robust to

inaccurate modelling due to the large value of K_C (see Section 8.6). Decreasing α causes an overshoot in the setpoint response but increases the load disturbance rejection capability.

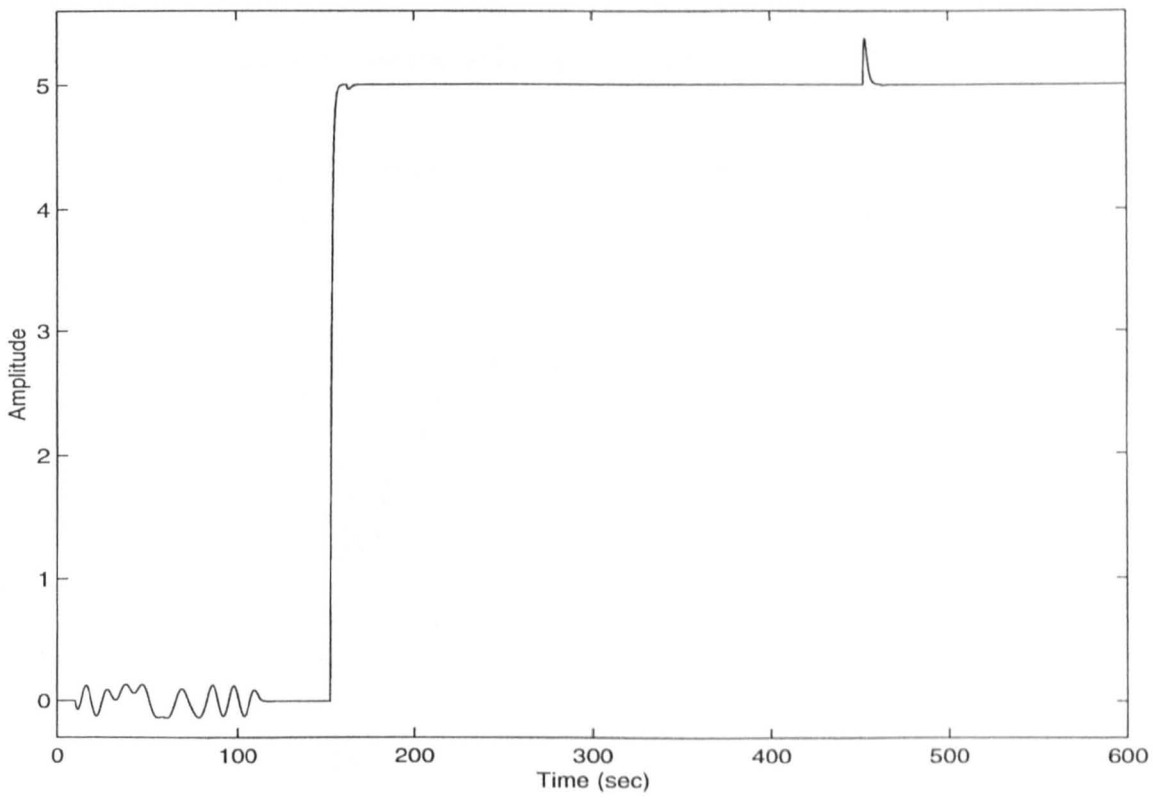

Figure 8.5. Process J : Tuning phase, step response and load response using PI controller with Smith predictor. The tuning rules of (8.7) with $\alpha = 1$ were used.

The response using PI control only (without the Smith predictor) is shown in Figure 8.6. For this, Ziegler-Nichols settings [90] were used.

$$K_C = 0.45K_U \tag{8.10A}$$

$$T_I = T_U/1.2 \tag{8.10B}$$

giving $K_C = 0.47$ and $T_I = 18.3$s. Note that T_I is now much larger than before due to the fact that the dead time is not compensated.

It may be seen that the output does settle out to the correct value, but only slowly. The discontinuities at $t = 152$s in response to the setpoint change and at $t = 452$s in response

to the load disturbance change both occur 10s after the respective changes are made, and are due to the changes being reflected round the loop after the dead time. Subsequent discontinuities at integer multiples of the dead time get progressively smaller. (The above phenomenon was also observed when a Smith predictor was incorporated into the control loop, and when a Dahlin controller was used. However, the amplitudes of these discontinuities were much smaller and they were hence less obvious.)

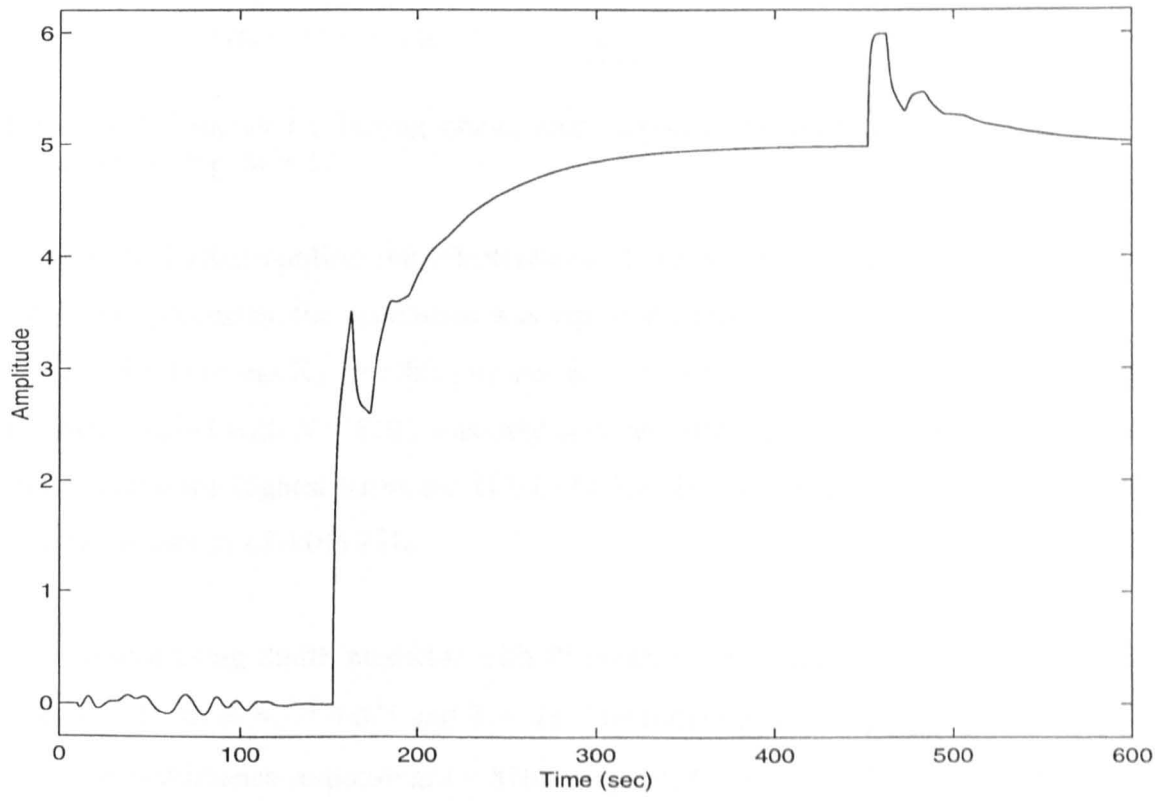

Figure 8.6. Process J : Tuning phase, step response and load response using PI controller without Smith predictor. Ziegler-Nichols tuning rules (equations (8.10)) were used.

In Figure 8.7, a Dahlin controller was used with the discrete controller having a sampling interval of 0.04s, making it the same as that for the rest of the control loop; thus giving $N_R = 250$. With T_P remaining at 1 second, $M = 0.0392$. The ratio Q/M was set at 2 giving $Q = 0.0784$, which corresponds to $T_{CL} = 0.490$s. (A smaller value of Q/M gives a slower response while a larger value results in a more oscillatory response.) From Figure 8.7, it may be seen that the step response is good; but that the response to a load disturbance is poor, having a large amplitude.

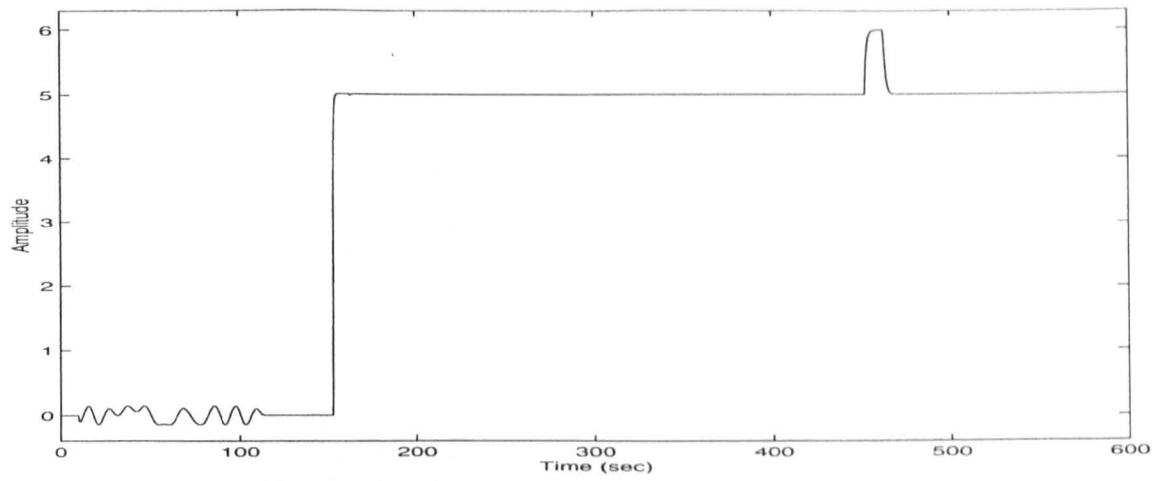

Figure 8.7. Process J : Tuning phase, step response and load response using Dahlin controller with $Q/M = 2$.

In order to further confirm the effectiveness of the Smith predictor with PI control on first order processes, the simulation was repeated using a process with $K_P = 1$, $T_P = 2$s and $L = 40$s (Process K). For this process, $K_U = 1.01$ and $T_U = 84.0$s. An eight-harmonic multisine signal with $N = 8192$ was used and the sampling frequency was set at 25 Hz, thus placing the highest harmonic at 0.0244 Hz. This is reasonably close to twice the critical frequency of 0.0119 Hz.

The control using Smith predictor with PI control (tuned according to equations (8.8) with $\beta = 1$) gives $K_C = 0.075$ and $T_I = 2$s. The tuning phase, step response at $t = 450$s and load disturbance response at $t = 810$s are plotted in Figure 8.8. The responses using PI control only (with $K_C = 0.45$ and $T_I = 70.0$s, tuned according to equations (8.10)) and using the Dahlin controller (with $M = 0.0198$, $Q = 0.0396$ and $N_R = 1000$, tuned according to equation (8.9)) are also illustrated for comparison.

From the figure, a good response is observed using the Smith predictor with PI control. The response is slower than that in Figure 8.4 due to the increased dead time. It can be observed that the response using PI control alone is extremely sluggish. With the Dahlin controller, the response to a change in setpoint is good, but the response to a load disturbance has a large amplitude. Thus, it is again shown that the Smith predictor with PI control is the best controller to use for a first order plus dead time process.

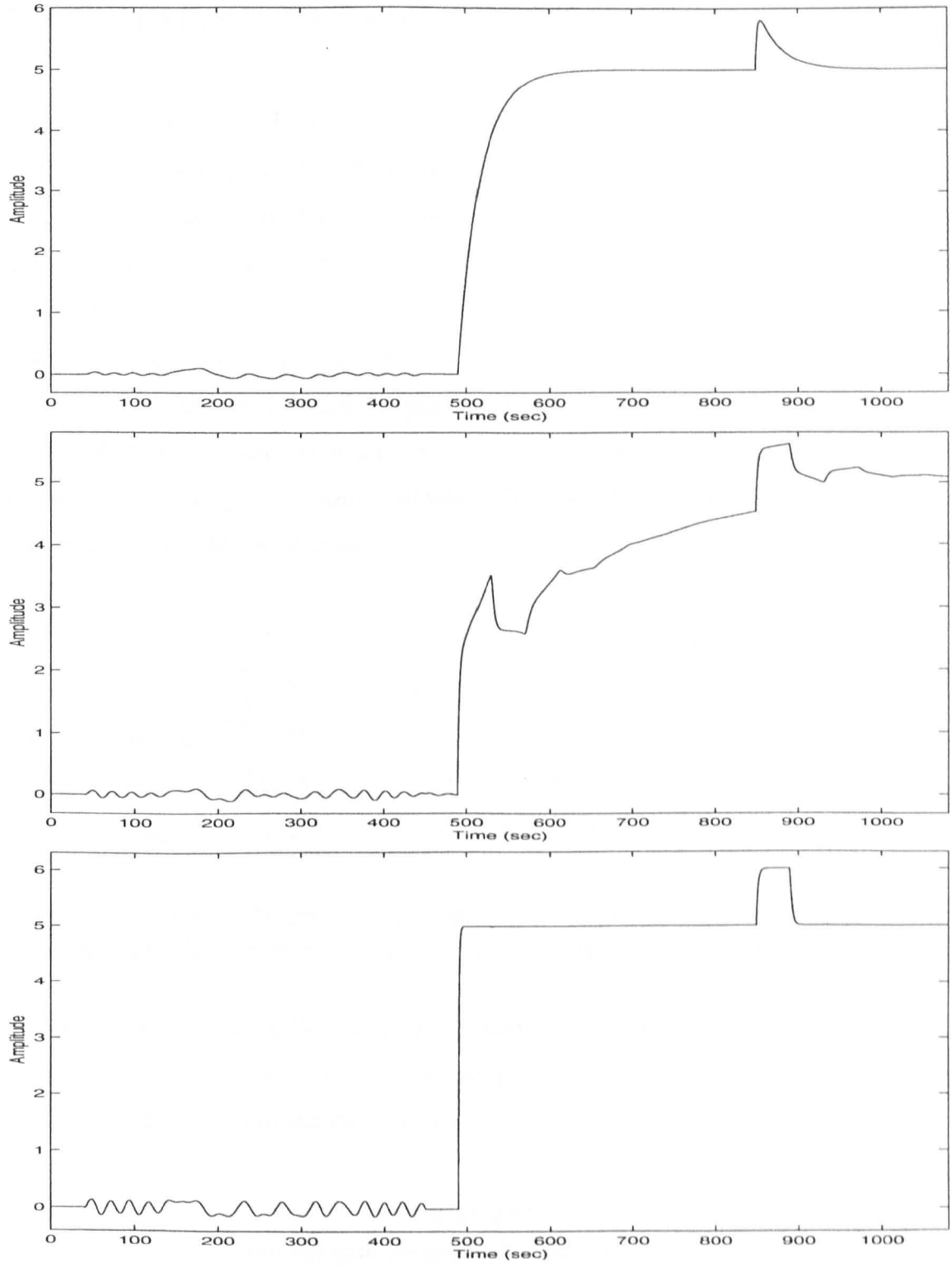

Figure 8.8. Process K : Tuning phase, step response and load response for a first order plus dead time process using three different control strategies.
Top : Smith predictor with PI control.
Middle : PI control only.
Bottom : Dahlin controller.

8.6. Controller Robustness

The Smith predictor with PI control is robust in the sense that if the initial parameter estimates are inaccurate, the subsequent estimates will improve in accuracy over the number of iterations, and the closed-loop will not become unstable. To illustrate this, consider again Process J (with $K_P = 1$, $T_P = 1$s and $L = 10$s), and suppose that the initial estimates for K_P, T_P and L are 0.75, 1.25s and 12.5s respectively (25% error in each estimate). The results shown in Table 8.2 are for the case where no noise was added to the system; it may be seen that the parameters quickly converge towards their correct values. With noise added, the values of the estimates again improved over the number of iterations but only up to a certain limit where these values then oscillated around the true values due to the effects of noise.

	K_P	T_P(s)	L (s)	K_C	T_I(s)
Theoretical	1.00	1.00	10.00	0.30	1.00
Initial estimates	0.75	1.25	12.50	0.32	1.25
After 1 iteration	0.94	0.71	10.36	0.31	0.71
After 2 iterations	1.00	1.01	10.00	0.30	1.01

Table 8.2. Process J : Actual and estimated values for process and controller parameters using Smith predictor with PI control, for the situation with no noise added. $\beta = 1$.

Another way to compare the robustness of controllers is to use a robustness plot [128] which shows the sensitivity of the closed-loop to mismatches in process gain or process dead time. The gain ratio and the delay ratio are defined as

$$\text{Gain ratio} = \frac{\text{process gain}}{\text{process gain the controller was tuned for}} \tag{8.11}$$

$$\text{Delay ratio} = \frac{\text{process dead time}}{\text{process dead time the controller was tuned for}} \tag{8.12}$$

The robustness plot (delay ratio versus gain ratio) for Process J using a Smith predictor with PI control is shown in Figure 8.9. For the controller tuned using equations (8.7A) and (8.7B), the region to the right-hand side of the dash-dot line is unstable; similarly, the right hand side of the dashed line indicates instability for a controller tuned using equations (8.8A) and (8.8B). It is also preferred that such a line does not cross into the enclosed window (shown as the solid line), which comprises all combinations of delay ratio and gain ratio within a factor of two in either direction. It can be seen in Figure 8.9 that the controller tuned using equations (8.8) is much more robust than that tuned using equations (8.7). It is common that controllers with dead time compensator will cross into the enclosed window [116]. However, due to the accurate identification method, this should not pose any problems if the region where the crossing occurs is small. The point at (1,1) is marked with a cross to indicate the ideal situation of accurate modelling.

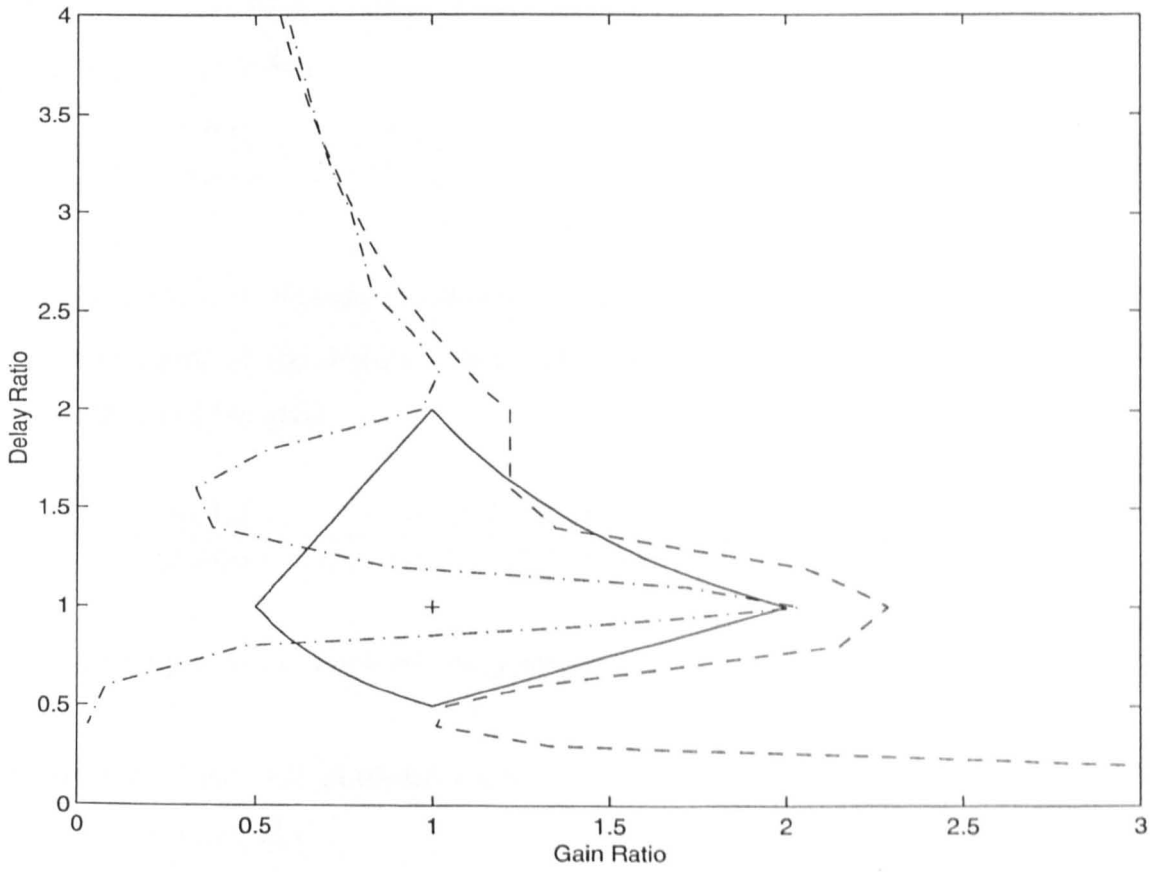

Figure 8.9. Robustness plot of Smith predictor with PI control for Process J. Solid line: Gain and delay ratios of a factor of two in either direction; Dashed line: Controller tuned using (8.8) with $\beta = 1$; Dash-dot line: Controller tuned using (8.7) with $\alpha = 1$.

8.7 Control of a Second Order Process

8.7.1 PI Control

Consider a second order plus dead time process of the form

$$H(\mathrm{s}) = \frac{K_P}{(\mathrm{s}T_P + 1)^2} \mathrm{e}^{-\mathrm{s}L} \tag{8.13}$$

K_P, T_P and L can be calculated using least squares from

$$K_P - |H|\omega^2 T_P^2 = |H| \tag{8.14}$$

$$-\omega L - 2\tan^{-1}\omega T_P = \angle H \tag{8.15}$$

In [125], the controller is designed such that the closed-loop response, compensated for dead time, is of the form

$$\frac{C(\mathrm{s})G(\mathrm{s})}{1 + C(\mathrm{s})G(\mathrm{s})} = \frac{\omega_n^{\,2}}{\mathrm{s}^2 + 2\zeta\omega_n \mathrm{s} + \omega_n^{\,2}} \tag{8.16}$$

assuming accurate process modelling. ω_n and ζ are the natural frequency and the damping factor of the desired closed-loop response respectively. Expanding the left-hand side of (8.16) gives

$$\frac{C(\mathrm{s})G(\mathrm{s})}{1 + C(\mathrm{s})G(\mathrm{s})} = \frac{sK_PK_CT_I + K_PK_C}{(sT_P + 1)^2 sT_I + sK_PK_CT_I + K_PK_C} = \frac{\frac{K_PK_C}{T_P^2}}{s^2 + \frac{s}{T_P} + \frac{K_PK_C}{T_P^2}} \tag{8.17}$$

if T_I is set equal to T_P, which was suggested in [125].

A possible solution can be obtained with

$$K_C = 1/(4\zeta^2 K_P)$$

$$T_I = T_P \tag{8.18}$$

leaving ζ as the only parameter to be tuned, which is chosen in the range 0.5 - 1 [125].

This controller was found to be reasonably robust. However, the robustness can again be increased by incorporating the idea of setting the controller gain to be inversely proportional to the dead time of the process [107]. It was proposed in [107] that a PID controller should be used and that the controller should incorporate the inverse of the dead time free part of the process model, which can be traced back to optimal control [129]. However, a PI controller which is well-tuned was found to be sufficient and the tuning rules were set the same as those in equations (8.8) for a first order process. Increasing K_C increases the overshoot due to a step response while decreasing K_C gives a slower response. Decreasing T_I improves the speed of response but increases the amplitude of the response due to a load disturbance.

8.7.2 Simulation Results

8.7.2.1 Parameter Estimation

In order to allow direct comparison with the first order process, a simulation was carried out for a second order process (Process L) with $K_P = 1$, $T_P = 1$s and $L = 10$s. For this process, the critical gain $K_U = 1.07$ and the critical period $T_U = 23.9$s. A multisine signal with $N = 2048$ was used, with the sampling frequency set at 22.2 Hz. This placed the highest harmonic at 0.0868 Hz, which is reasonably close to twice the critical frequency of 0.0418 Hz.

The resulting estimates of the process parameters K_P, T_P and L, and the controller parameters K_C and T_I (tuned according to equations (8.8)) are given in Table 8.3 for the cases with no noise added, and a Gaussian noise with SNR of 10dB and 15dB added. There is close match between theoretical values and those obtained through simulation. The gain response and phase response are shown in Figure 8.10, for the case with an SNR of 10dB.

	K_P	T_P(s)	L (s)	K_C	T_I(s)
Theoretical	1.00	1.00	10.00	0.30	1.00
No noise	1.00	1.00	10.00	0.30	1.00
SNR of 15dB	1.00	0.98	10.02	0.30	0.98
SNR of 10dB	1.00	0.98	10.02	0.30	0.98

Table 8.3. Process L : Actual and estimated values for process and controller parameters using Smith predictor with PI control. $\beta = 1$.

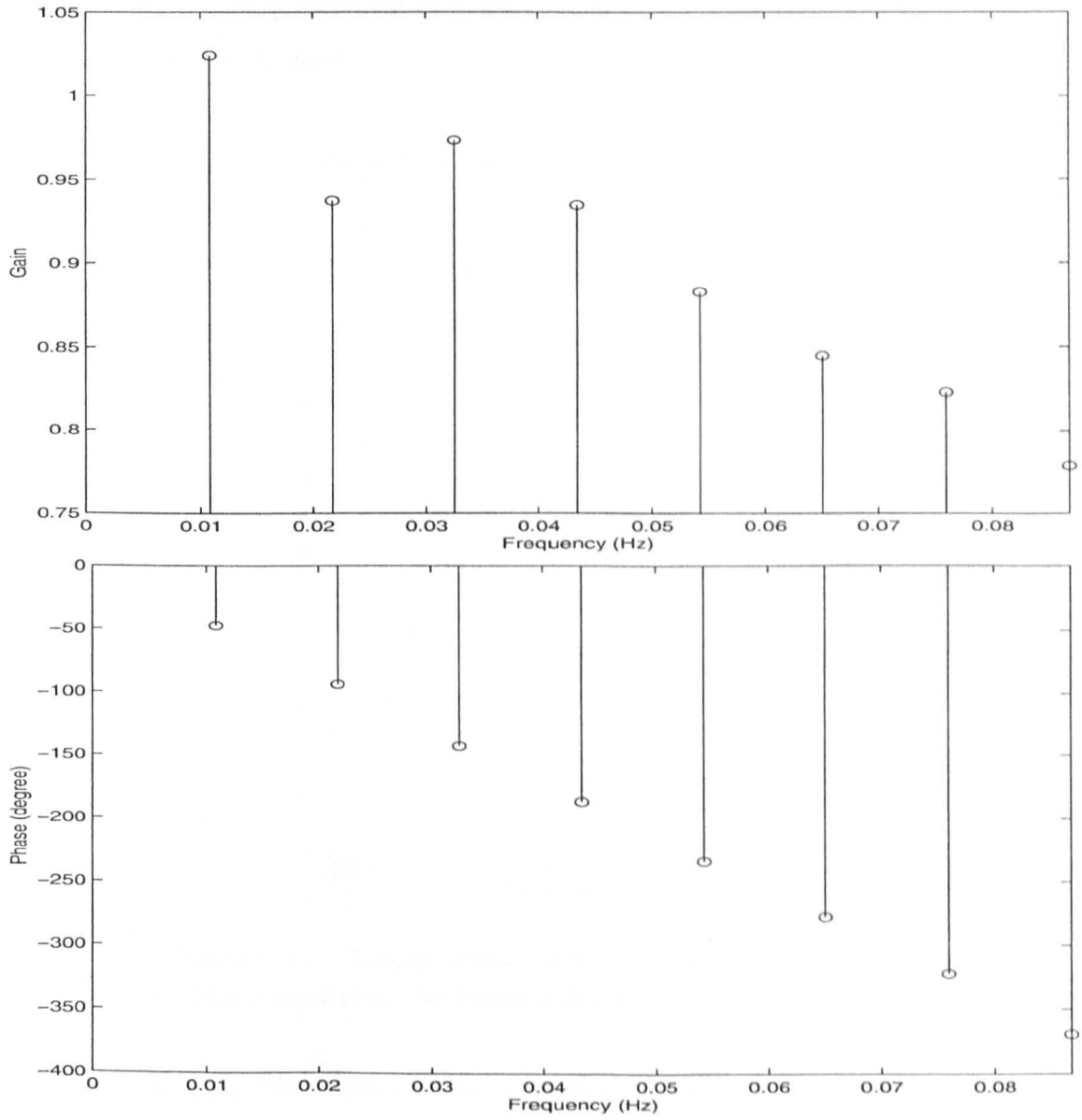

Figure 8.10. Gain response (top) and phase response (bottom) estimates for Process L with an SNR of 10dB.

8.7.2.2 Controller Responses

In Figures 8.11 to 8.13, the responses using different controllers in the noiseless case are shown. In each case, the tuning phase is carried out between $t = 0$ and $t = 115$s. A setpoint change of amplitude 5 is introduced at $t = 160$s and a load disturbance of amplitude 1 is then introduced at $t = 498$s.

For Figure 8.11, a Smith predictor with PI control was used, with $\beta = 1$. As for Process J, the output quickly settles to the correct steady-state value and the response to the load disturbance is good.

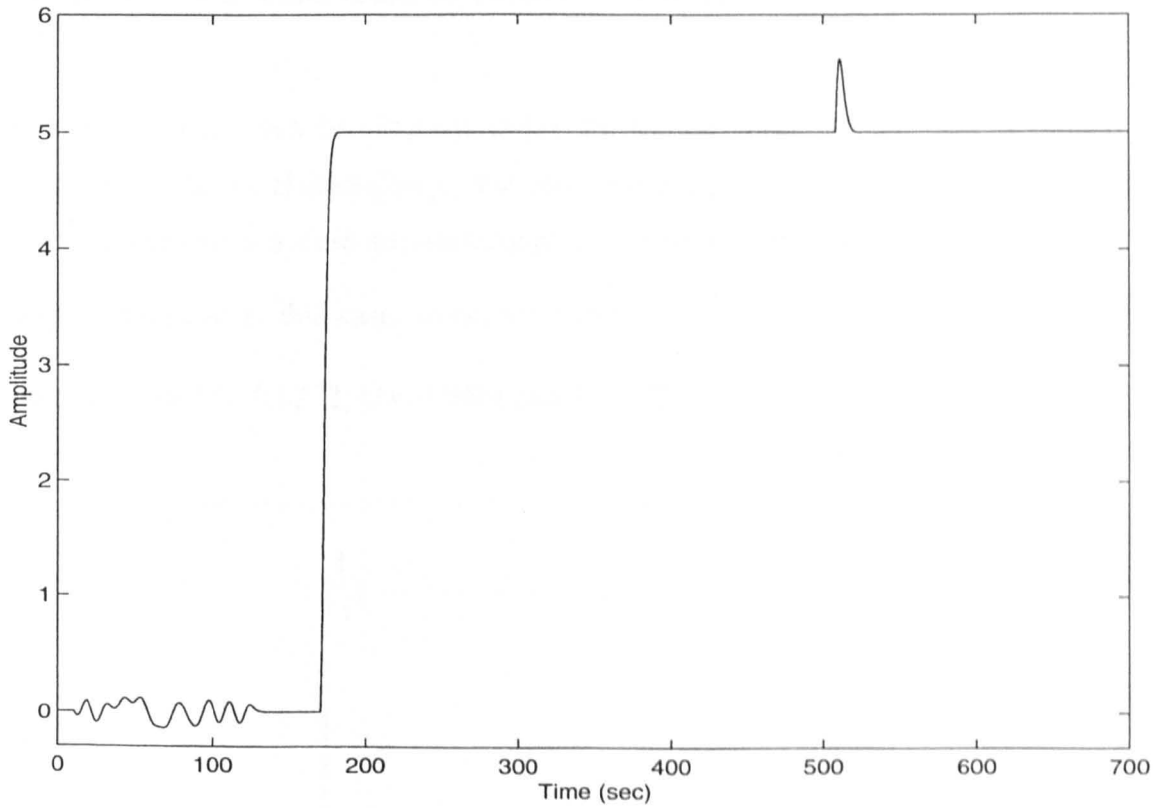

Figure 8.11. Process L : Tuning phase, step response and load response using PI controller with Smith predictor. The tuning rules of (8.8) with $\beta = 1$ were used.

The response using PI control without a Smith predictor is shown in Figure 8.12; this has $K_C = 0.48$ and $T_I = 19.9$s set according to equations (8.10A) and (8.10B). The response is seen to be very slow.

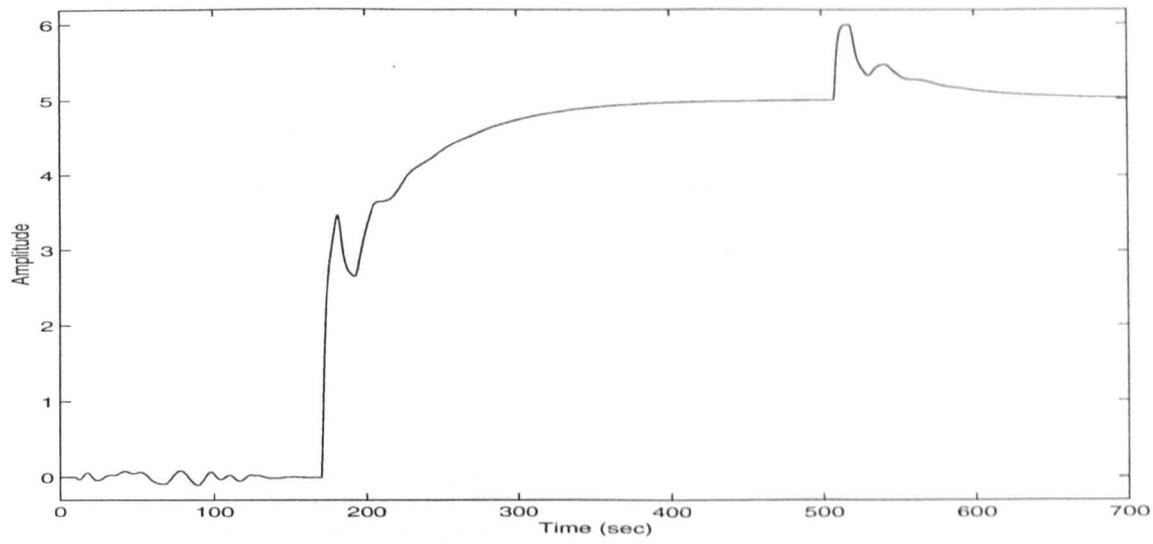

Figure 8.12. Process L : Tuning phase, step response and load response using PI controller without Smith predictor. Ziegler-Nichols tuning rules (equations (8.10)) were used.

A faster response can be obtained using the Dahlin controller and this is plotted in Figure 8.13. In the Dahlin design, the second order process is approximated by a first order process with a time constant equal to the sum of the two time constants in the original process; in this case, therefore, $H(s) = \dfrac{1}{2s + 1} e^{-10s}$. The parameters for the controller are $M = 0.0222$, $Q = 0.0444$ and $N_R = 222$.

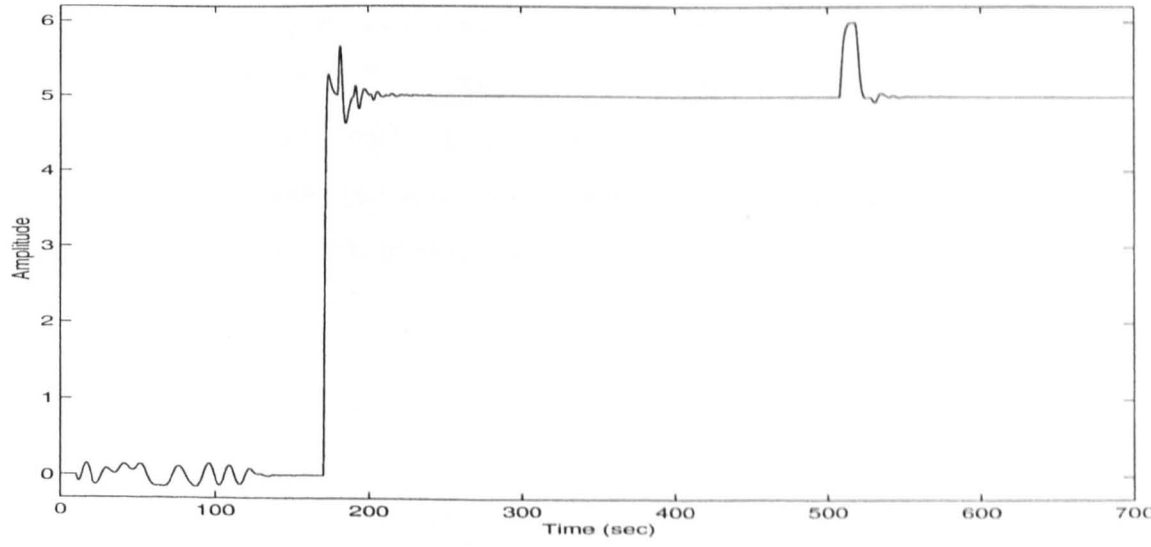

Figure 8.13. Process L : Tuning phase, step response and load response using Dahlin controller with $Q/M = 2$.

It may be seen from Figure 8.13 that the response to a change in setpoint is very oscillatory and requires a longer time to settle down to its steady-state value compared with that of the Smith predictor with PI control (shown in Figure 8.11). Similarly, the response of the former to a load disturbance is worse, having a larger amplitude and a longer settling time. It should be noted that the Dahlin algorithm is designed specifically for a first order process and hence does not perform equally well on a second order process.

A further experiment was conducted using a process with $K_P = 1$, $T_P = 2$s and $L = 40$s (Process M) in order to allow comparison with a first order process with similar process parameters (Process K). For this process, $K_U = 1.02$ and $T_U = 87.9$s. An eight-harmonic multisine signal with $N = 8192$ was used and the sampling frequency was set at 23.3 Hz, thus placing the highest harmonic at 0.0227 Hz, which is reasonably close to twice the critical frequency of 0.0114 Hz.

The PI control with Smith predictor (tuned according to equations (8.8) with $\beta = 1$) has parameters $K_C = 0.075$ and $T_I = 2$s. The tuning phase, step response at $t = 483$s and load response at $t = 870$s is shown in Figure 8.14. The response is seen to be reasonably fast and smooth. The responses using PI control only (with $K_C = 0.46$ and $T_I = 73.2$s) and using the Dahlin controller (with $M = 0.0213$, $Q = 0.0425$ and $N_R = 930$), are also shown for comparison. Using PI control alone, the response is very slow and quite oscillatory. With the Dahlin controller, the step response is extremely oscillatory. It can be seen from the figure that the Smith predictor with PI control is the optimal control technique to use (among the three strategies considered) in terms of the speed and smoothness of response for a second order process with significant dead time.

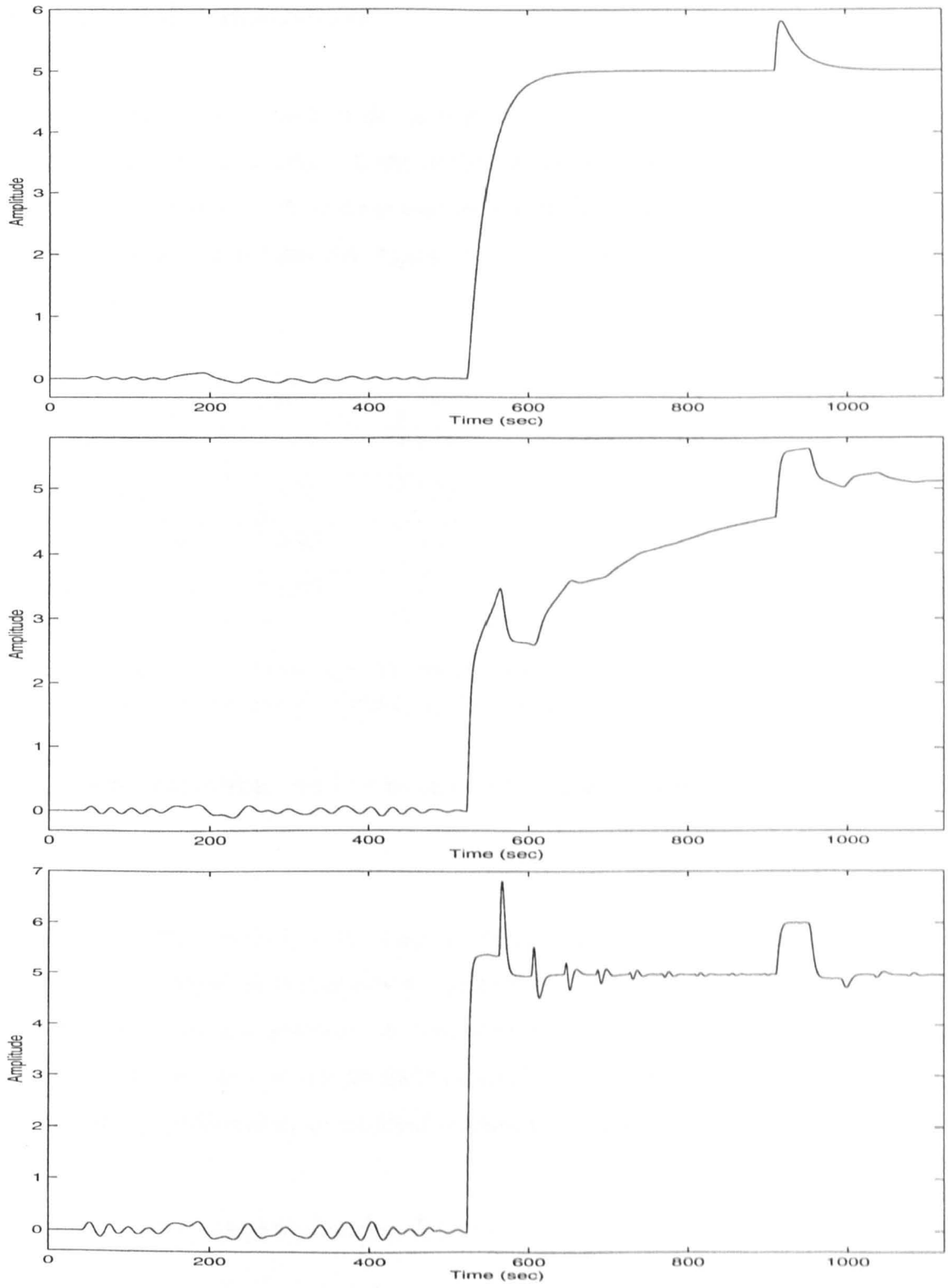

Figure 8.14. Process M : Tuning phase, step response and load response for a second order plus dead time process using three different control strategies.
Top : Smith predictor with PI control.
Middle : PI control only.
Bottom : Dahlin controller.

8.7.3 Controller Robustness

The robustness of the Smith predictor with PI control is illustrated in Table 8.4 for Process L with no noise added. If the initial estimates for K_P, T_P and L are 0.75, 1.25s and 12.5s respectively (25% error in each estimate), the values after the first and second iterations are shown in Table 8.4. Again, the parameters converge quickly towards their correct values.

	K_P	T_P(s)	L (s)	K_C	T_I(s)
Theoretical	1.00	1.00	10.00	0.30	1.00
Initial estimates	0.75	1.25	12.50	0.32	1.25
After 1 iteration	0.92	0.51	10.77	0.30	0.51
After 2 iterations	1.00	0.97	9.54	0.31	0.97

Table 8.4. Process L : Actual and estimated values for process and controller parameters using Smith predictor with PI control, for the situation with no noise added. $\beta = 1$.

8.7.4 Identification and Control of Second Order Underdamped Processes

In the approach described in this chapter, the identification of a second order process assumes that the process is critically damped (from (8.13)). For an overdamped process, this does not pose any problems in control since the process itself is not oscillatory. However, this is not true for an underdamped process. It is therefore of interest to investigate the performance of the identification and control strategy for such a process.

A process with the transfer function $\overline{H_1}(s) = \frac{1}{s^2 + 2s + 2} e^{-10s}$ (Process N) was simulated. This has a damping factor of 0.707, an undamped natural frequency of 1.41 rad s^{-1} and a critical frequency of 0.0454 Hz. Identification using a multisine with $N = 2048$ and a sampling frequency of 23.3 Hz gave the process model as $H_1(s) = \frac{0.500}{(0.124s + 1)^2} e^{-10.8s}$,

which is considerably different from the actual process chiefly because the fitted model was constrained to a damping factor of unity.

The control of the process using a Smith predictor with PI control is shown in Figure 8.15. $K_C = 0.139$ and $T_I = 0.124$s with $\beta = 4$ since the process is quite underdamped. The tuning phase is carried out between $t = 0$ and $t = 110$s. The response to a step input at $t = 153$s is rapid with very little overshoot and the response to a load disturbance at $t = 368$s is also seen to be excellent, despite the remarkable difference between the actual process and its model.

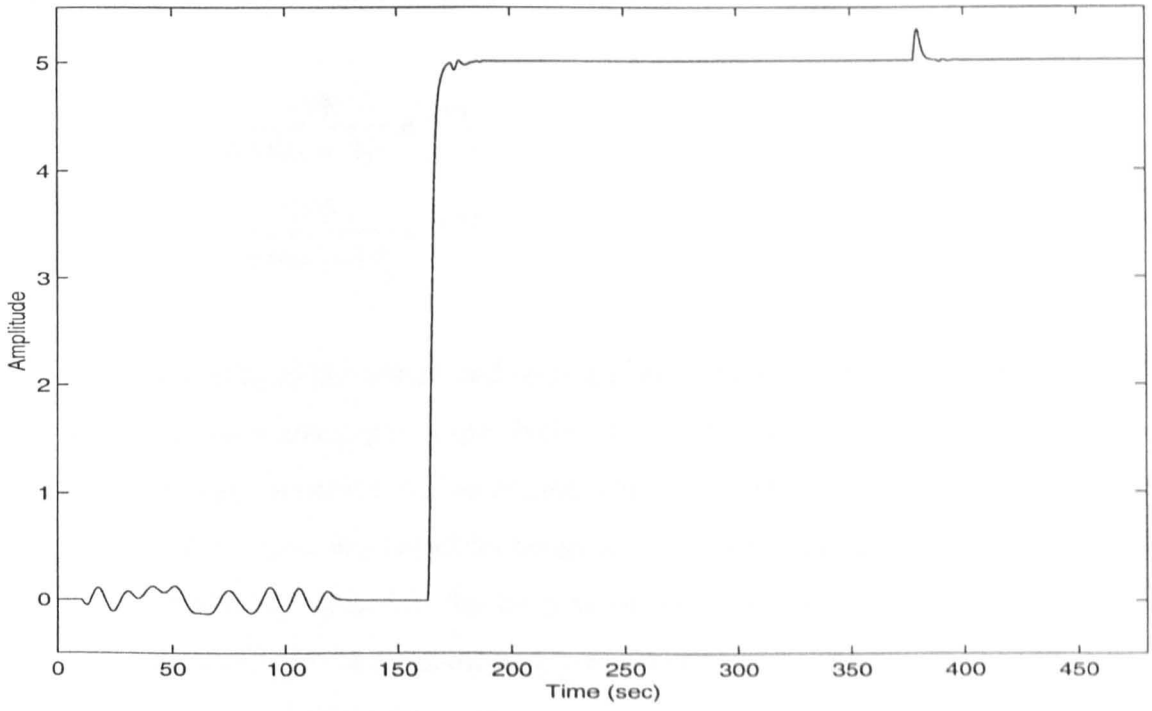

Figure 8.15. Process N : Tuning phase, step response and load response using PI controller with Smith predictor. The tuning rules of (8.8) with $\beta = 4$ were used.

8.8 Reduced Order Modelling

Many high order or non-minimum phase processes can be modelled using a lower order model with dead time [130]. It has been shown through extensive simulations [131, 132] that a second order model is adequate in most cases.

8.8.1 High Order Process

Two processes with transfer functions

$$\overline{H_2}(s) = \frac{1}{(s+1)^{10}} \qquad \text{and}$$

$$\overline{H_3}(s) = \frac{1}{(s+1)^{30}}$$

were considered. Identification using multisine signal (carried out assuming that the processes were second order) gave the reduced order models as

$$H_2(s) = \frac{1.08}{(3.01s+1)^2} e^{-5.27s} \qquad \text{and}$$

$$H_3(s) = \frac{1.03}{(4.66s+1)^2} e^{-22.01s}$$

The Nyquist curves of the actual and reduced order models are shown in Figures 8.16 and 8.18 for $H_2(s)$ and $H_3(s)$ respectively. Note that these are closely matched thus proving that the identification technique can be used effectively for high order processes. The response to a setpoint change at t = 142s and a load disturbance change at t = 346s is shown in Figure 8.17 for the process with transfer function $H_2(s)$. K_C = 0.53 and T_I = 3.01s (with β = 1) set according to equations (8.8). (The tuning phase is carried out from t = 0 to t = 102s.) The response to a setpoint change at t = 337s and a load disturbance change t = 607s is shown in Figure 8.19 for the process with transfer function $H_3(s)$. The controller parameters are K_C = 0.13 and T_I = 4.66s (with β = 1). (The tuning phase is carried out from t = 0 to t = 307s.) It can be seen that the controller gives a reasonably good setpoint and load disturbance control in both cases (Figures 8.17 and 8.19).

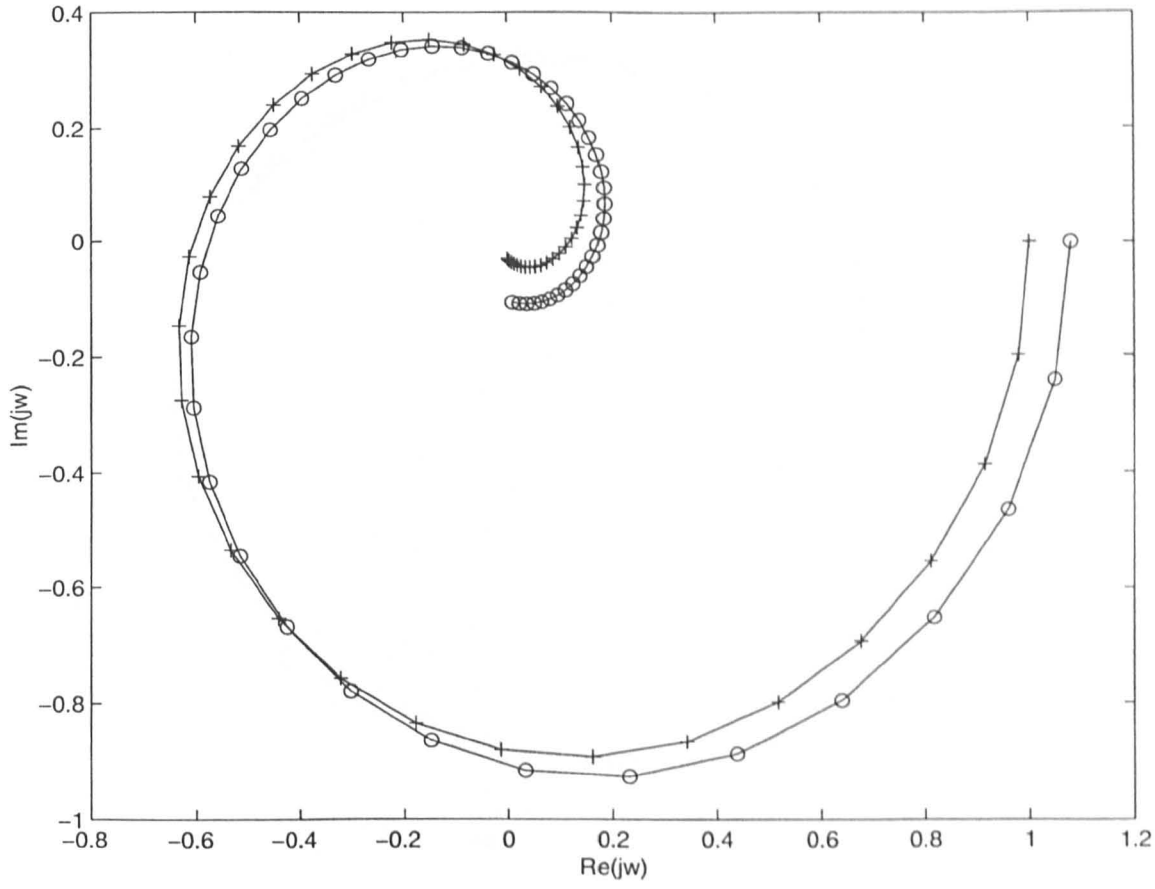

Figure 8.16. Nyquist curves for process with transfer function $H_2(s)$. Line with plusses: Actual process; Line with circles: Reduced order model.

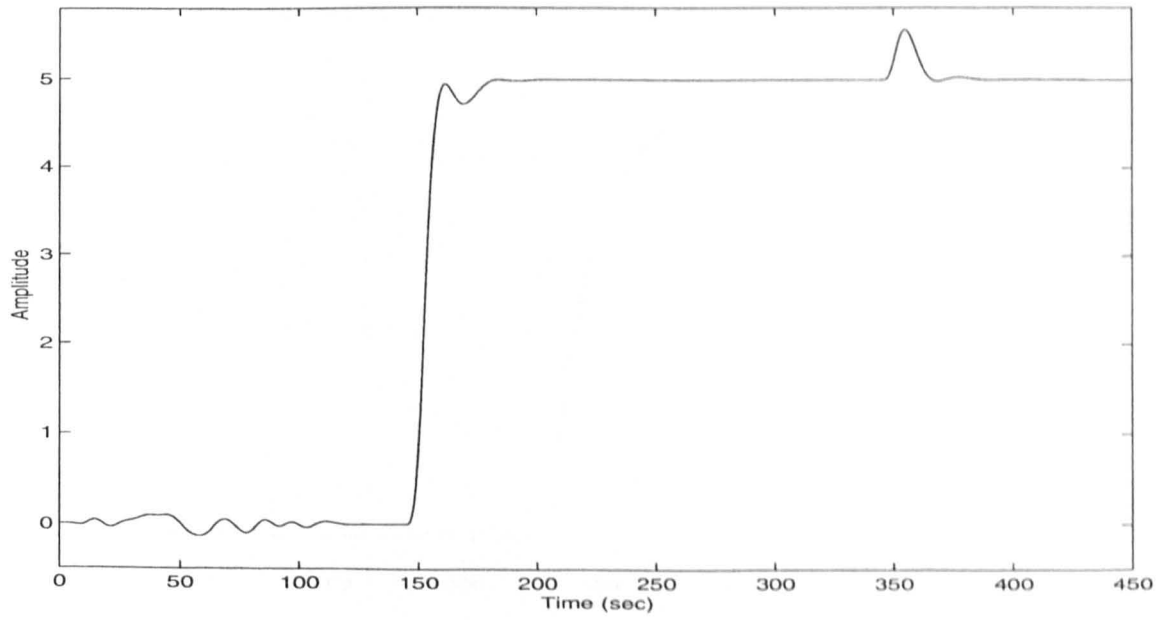

Figure 8.17. Tuning phase, step and load response using reduced order modelling for process with transfer function $H_2(s)$.

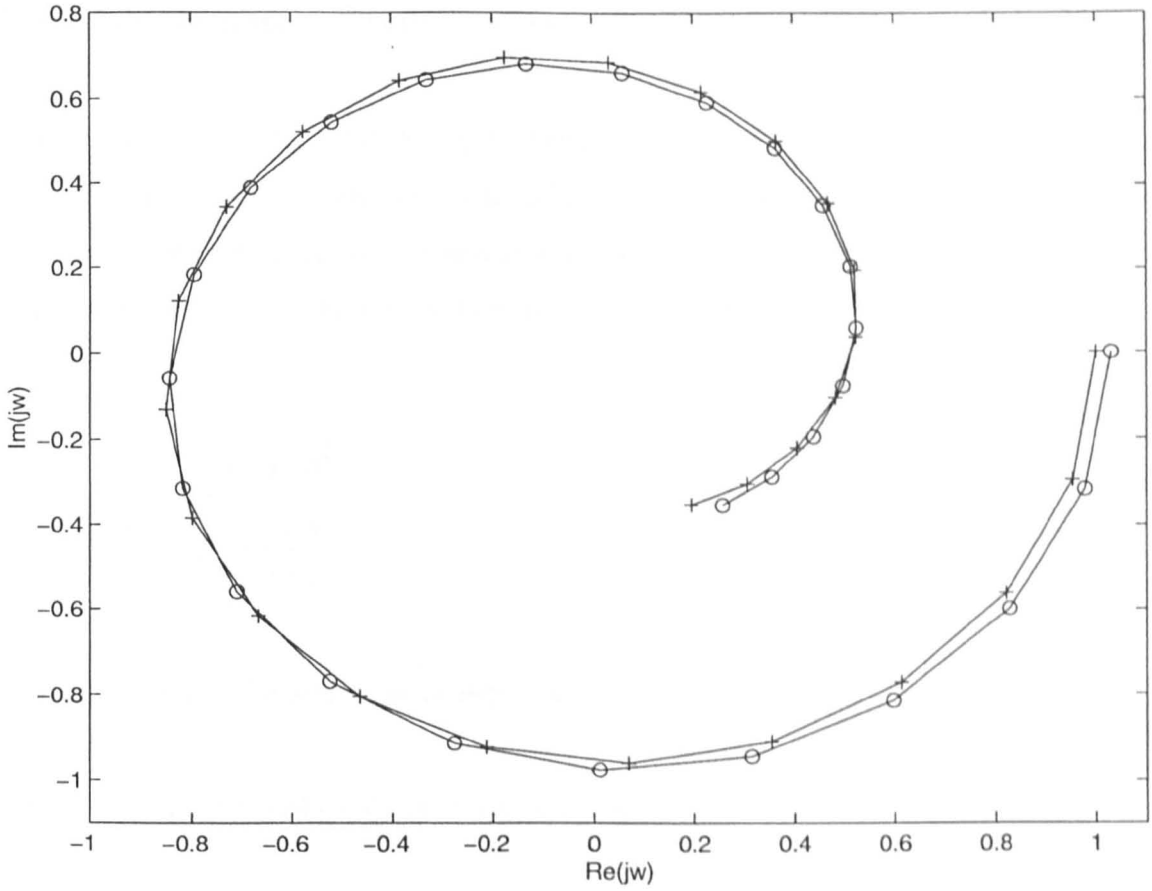

Figure 8.18. Nyquist curves for process with transfer function $H_3(s)$. Line with plusses: Actual process; Line with circles: Reduced order model.

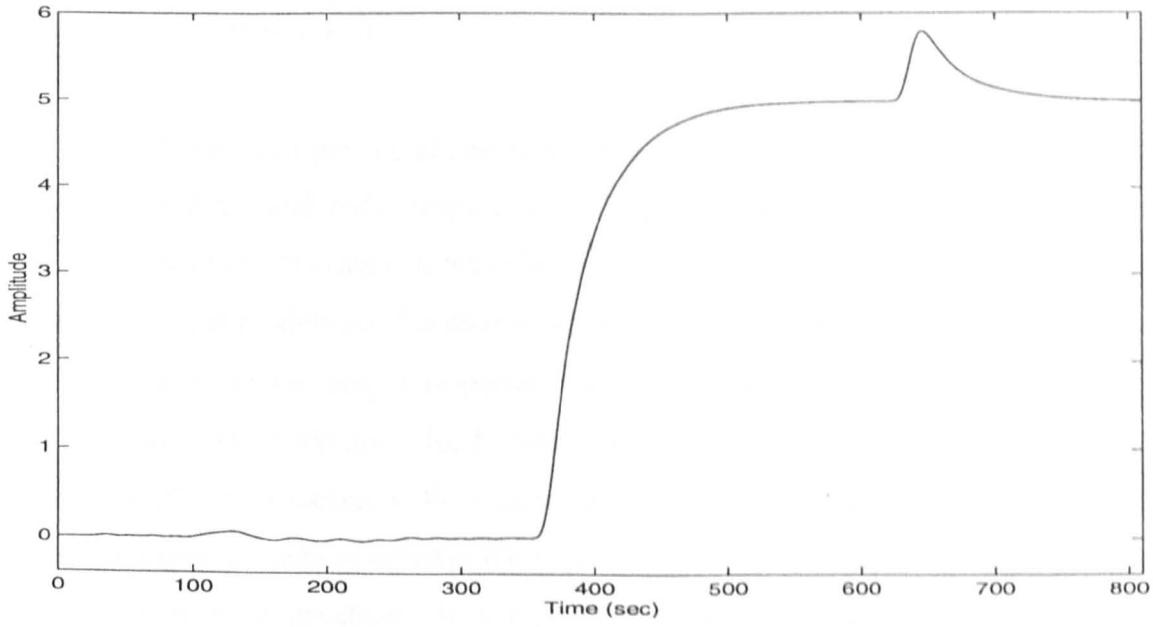

Figure 8.19. Tuning phase, step and load response using reduced order modelling for process with transfer function $H_3(s)$.

8.8.2 Non-minimum Phase Process

It is shown in [131, 132] that non-minimum phase processes can be modelled using second order processes with dead time. It is then of interest to evaluate the effectiveness of the proposed identification technique and subsequently, the controller design for this class of processes. Two processes with transfer functions

$$\overline{H_4}(s) = \frac{1 - s}{(s + 1)^3} \qquad \text{and}$$

$$\overline{H_5}(s) = \frac{1 - 1.5s}{(s + 1)^3}$$

were considered. (These processes were also considered in [125].)

These could be reduced to second order processes as below

$$H_4(s) = \frac{0.86}{(0.83s + 1)^2} e^{-1.23s} \qquad \text{and}$$

$$H_5(s) = \frac{0.91}{(0.66s + 1)^2} e^{-1.42s}$$

The Nyquist curves of the actual and reduced order models are shown in Figures 8.20 and 8.22 for $H_4(s)$ and $H_5(s)$ respectively. These are less well matched compared to those for high order processes, as was also found in [125]. Due to the small dead time in the reduced order models and the relatively poorer matching in the Nyquist plane, β was set to 20 to avoid the output response from becoming unstable. The response to a setpoint change at t = 66s and a load disturbance change at t = 226s is shown in Figures 8.21 and 8.23 for processes with transfer functions $H_4(s)$ and $H_5(s)$ respectively. The initial transients are seen to move in the opposite direction, and this is a characteristic of non-minimum phase processes. It is clear from these figures that the controller still performs well for this particular class of processes.

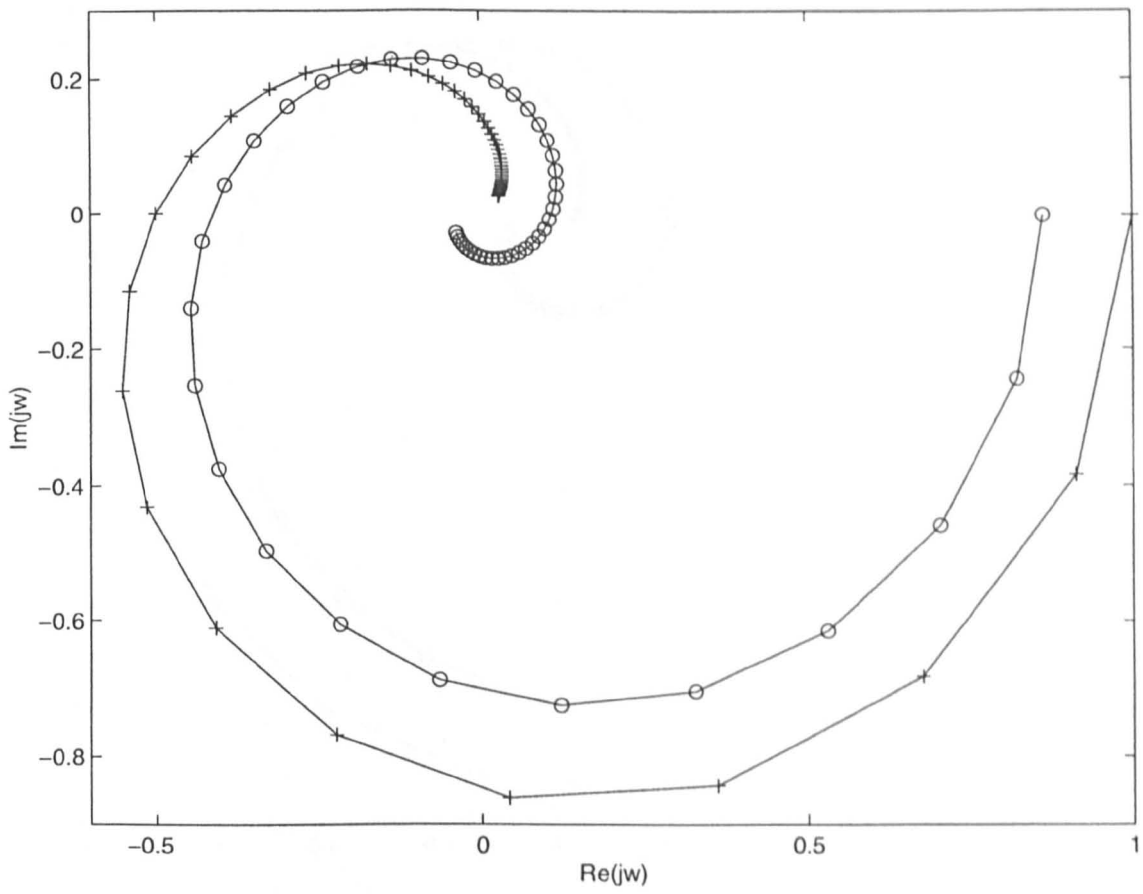

Figure 8.20. Nyquist curves for process with transfer function $H_4(s)$. Line with plusses: Actual process; Line with circles: Reduced order model.

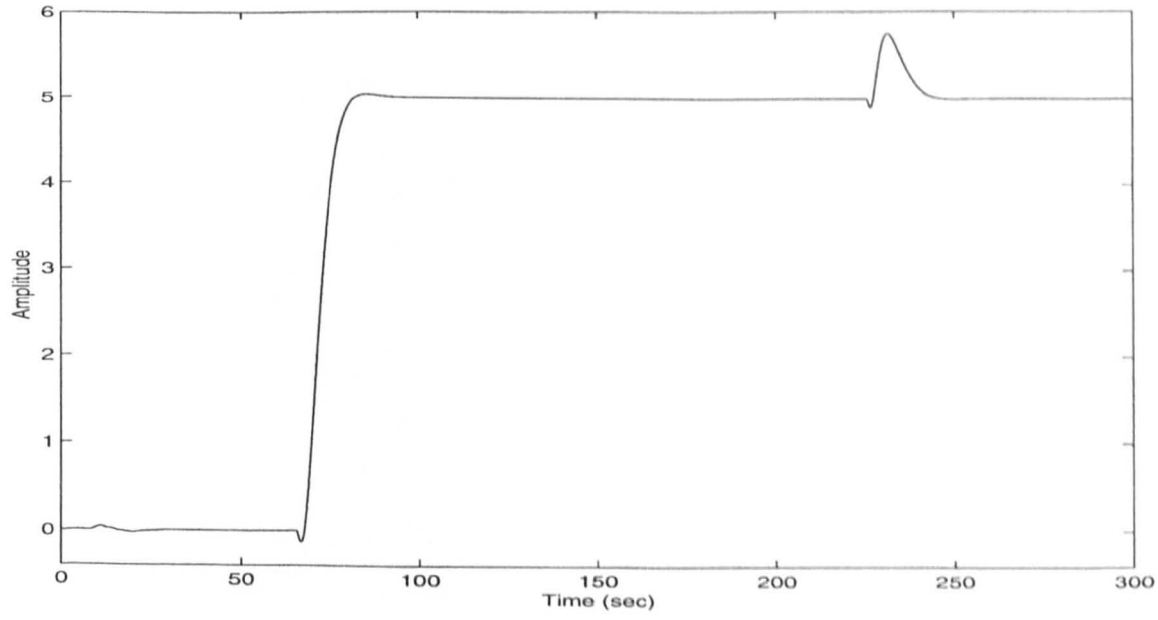

Figure 8.21. Tuning phase, step and load response using reduced order modelling for process with transfer function $H_4(s)$.

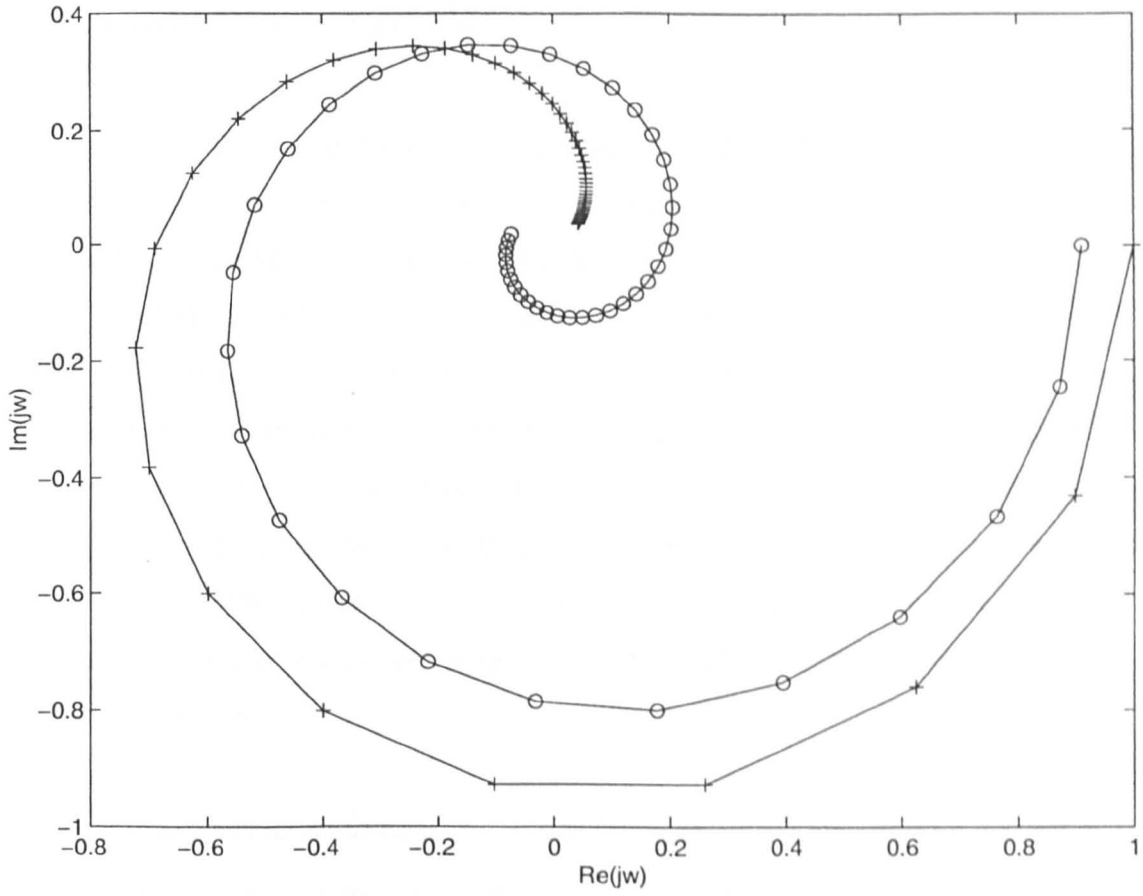

Figure 8.22. Nyquist curves for process with transfer function $H_5(s)$. Line with plusses: Actual process; Line with circles: Reduced order model.

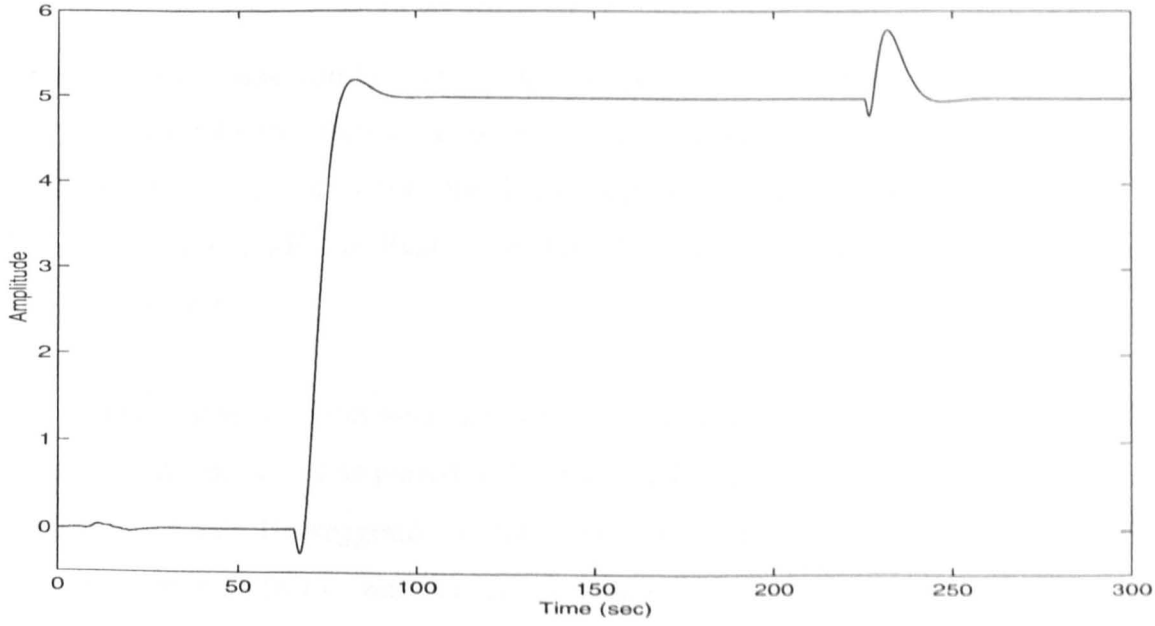

Figure 8.23. Tuning phase, step and load response using reduced order modelling for process with transfer function $H_5(s)$.

8.9 Application Example

To illustrate its practical nature, the approach was applied to a PT 326 hot-air flow device [133] available from Feedback Instruments Ltd. In this device, the air is drawn from the atmosphere by a centrifugal blower and is driven past a heater grid through a length of plastic tube back to the atmosphere. The temperature of the air is measured by a thermistor at the end of the tube. The system is approximately second order (though the actual order is not precisely known) as the high frequency roll-off of the magnitude frequency response is about 12dB per octave [134]. Although it is a laboratory scale process, it is highly representative of the type of problems which are encountered when designing a control system for a full scale process. It has a limited linear range, an internal noise source due to turbulence within the plastic tube and the temperature tends to drift in open-loop.

In the experiments reported here, the angular position of the blower inlet was set at 30°. At this setting, the air flow along the tube is relatively small, resulting in a significant dead time. (The same device was investigated in [100] but with a larger setting of the angular position of the blower inlet.)

The simulation was carried out using SIMULINK with the hot-air flow device being incorporated into the control loop using the Real-Time Windows Target [135] in MATLAB. (The data acquisition board used was the National Instruments PCI-6031E board [136].) The MATLAB Real-Time Workshop [137] was then used to execute the code in real time.

A multisine with $N = 256$ was used with the sampling frequency set at 80 Hz. The highest harmonic was thus placed at 2.5 Hz which is approximately twice the Nyquist frequency of 1.23 Hz suggested by the manufacturer. Identification of a model of the form of equation (8.13) was carried out using least squares (equations (8.14) and (8.15)). Results from 25 experiments were averaged. The frequency response is plotted in Figure 8.24, from which it can be seen that the uncertainty regions are quite large.

(The uncertainty regions are larger than those shown in [100] due to the difference in the angular position of the blower inlet.) This shows that the system was very noisy and difficult to control, and this was made worse by the tendency of the system to drift.

The frequency domain identification using a second order model gave the system parameters as K_P = 1.30, T_P = 0.33s and L = 0.21s. In fact, when the non-parametric frequency response data was estimated using the ARX model in the System Identification Toolbox [45], the algorithm identified the poles as 0.974 ± 0.048j, with the sampling frequency scaled to 1 Hz. This is very close to the predicted double poles at 0.963 (corresponding to T_P = 0.33s) considering the fact that the closeness of the poles to the unit circle in the z-plane made the identification difficult. The system is therefore very slightly underdamped. The values of K_U and T_U were found to be 2.94 and 1.15s respectively. These were calculated by linear interpolation between successive points in the estimated frequency response in order to decrease the computational time; this is justified since the estimated frequency response function is smooth.

The Smith predictor with PI control has the parameters K_C = 5.49 and T_I = 0.33s. β was set to 2 in order to increase the closed-loop stability since the system was very noisy. The tuning phase was carried out from t = 0 to t = 4s. The response to a step input of 3 volts at t = 7.75s and to a load disturbance of 2 volts at t = 20.3s is plotted in Figure 8.25. The responses using PI control without Smith predictor (with K_C = 1.32 and T_I = 0.96s) and the Dahlin controller (with M = 0.0372, Q = 0.0744 and N_R = 17) are also shown for comparison. (Note that a slight drawback when using the first control strategy is that the response to the tuning signal has a larger amplitude (due to the larger gain) than when using the other two strategies.) The response using PI control only is quite oscillatory. With the Dahlin controller, the step response is less oscillatory compared to that using PI control only but it has the largest overshoot of the three controllers. The load response also has the largest amplitude of these three controllers. The Smith predictor with PI control clearly has the best performance with rapid step and load responses, and very little overshoot. This result is extremely interesting (and perhaps slightly surprising) considering that the ratio of L to T_P is not significantly large.

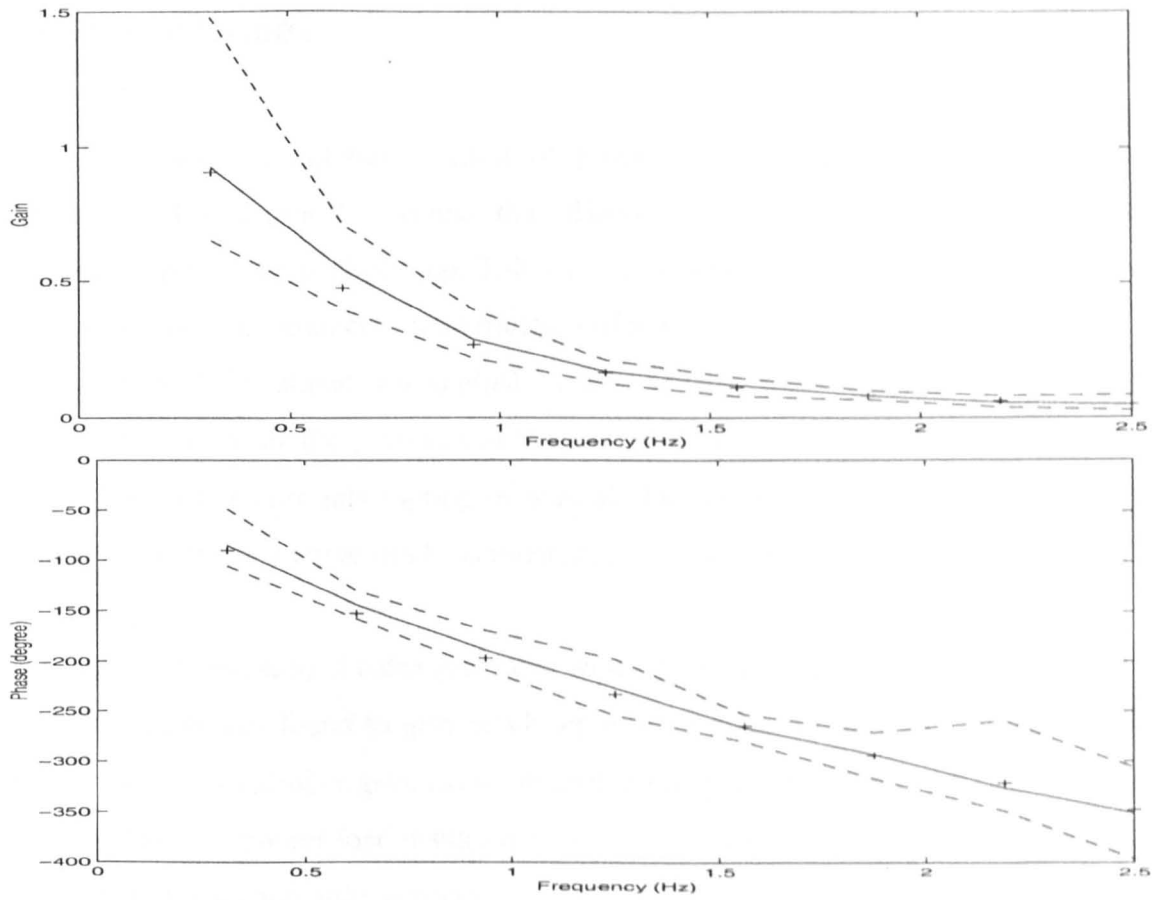

Figure 8.24. Frequency response estimates of the hot-air flow system. Solid line: Mean; Dashed lines: Uncertainty region (maximum and minimum values obtained from 25 experiments); Plusses: Second order model.

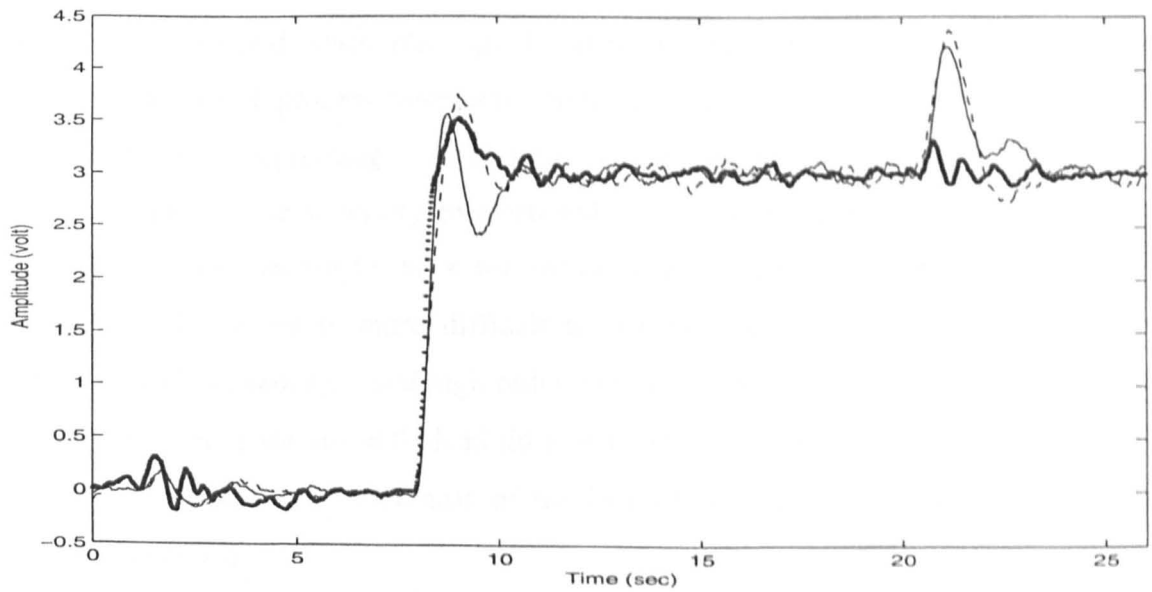

Figure 8.25. Response of a hot-air flow device. Dotted and thick line: Smith predictor with PI control; Solid line: PI control only; Dashed line: Dahlin controller.

8.10 Conclusions

In this chapter, the autotune control of processes with significant dead time was considered. The approach involves the addition of a small amplitude signal at the reference input of the control loop. This signal, provided it is persistently exciting, can be used for the non-parametric identification of a process, and in all the examples in this chapter, a multisine signal was applied. A least squares procedure can then be used to obtain an estimate of the process transfer function. An advantage of this technique is that it does not require interruption of normal closed-loop control. Additionally, many points on the Nyquist curve can be estimated simultaneously.

For first order and second order processes with significant dead time, a Smith predictor with PI control was found to give much better performance than a PI controller on its own. A Dahlin controller gave excellent setpoint response for first order processes, but at the expense of poorer load disturbance rejection capability. Its performance was less satisfactory for second order processes.

For the PI controller with Smith predictor, a simple set of tuning rules was proposed which can be used for both first order and second order processes. The controller is robust to noise and inaccurate initial parameter estimates. The sensitivity of the controller to model–process mismatches in the dead time (which is a common problem in controllers incorporating a dead time compensator) was reduced by setting the controller gain to be inversely proportional to the estimated dead time. The control strategy was also shown to work for second order underdamped processes (since an underdamped process is more difficult to control compared to an overdamped or critically damped process), and high order and non-minimum phase processes modelled as second order processes with dead time. An application example using a hot-air flow device illustrated the effectiveness of the identification procedure and the controller design in practice.

Chapter 9

Identification of Wiener-Hammerstein Models using Linear Interpolation in the Frequency Domain (LIFRED)

9.1 Abstract

A new method to identify the linear subsystems of a Wiener-Hammerstein model through the measurement of the second order Volterra kernel is proposed. This technique makes use of the symmetry properties of the Volterra kernel and assumes that the frequency response gain and phase between estimated points can be reasonably well approximated by a straight line. The signal applied for the identification is a multisine with properties of no interharmonic distortion. Several advantages of the proposed method over existing ones are discussed and two simulation examples are presented to illustrate the applicability of the technique. The method is also shown to be robust to noise and input signal distortion. Several criteria for model selection are discussed and applied to the simulation examples.

9.2 Introduction

Block-oriented models have been widely used to model nonlinear systems largely due to their simplicity. The basic building blocks for such models are the Wiener [11, 138] and Hammerstein [139] structures, shown in Figures 9.1(a) and 9.1(b) respectively. The identification of these structures was investigated in [18] using multi-level maximum length signals [8]. In [53], the identification of the Wiener model was compared using two different approaches, the first being the output error approach [54, 55], and the second being the internal error approach [56]. In the former approach, a linear model is estimated from the input-output data. The simulated output is then plotted against the measured output and the nonlinearity will show up on the plot if the linear estimate is reasonably accurate. The latter approach uses a finite impulse response filter and B-splines to describe the linear system and the inverse of the nonlinearity respectively. An example using an exponential nonlinearity was used in [53] to show that the internal error approach performs better in that particular case. The Wiener-Hammerstein model (shown in Figure 9.1(c)) has also been widely investigated in the literature. Identification techniques proposed include those using multisines with carefully chosen

harmonics to measure the Volterra kernels [140, 141], a series of large and small scale multisines [142], interpolation of the Volterra kernel using B-splines [143], and adaptive gradient search algorithms using Gaussian inputs [144]. The Hammerstein-Wiener structure (illustrated in Figure 9.1(d)) is much less well known and is effectively much more difficult to identify due to the presence of two blocks of nonlinearities. This problem has been considered in [145], where recursive least squares and singular value decomposition are applied in the identification process.

In this chapter, a new technique to identify the linear subsystems of a Wiener-Hammerstein model based on linear interpolation between measured points in the frequency domain is proposed. The analysis that follows assumes that the nonlinearity is quadratic, and that this is known *a priori*. However, the technique will also work if in addition to the assumed quadratic nonlinearity, there are odd order nonlinearities present. Two other pieces of *a priori* information are required and these are the approximate system bandwidth and the phase of the first linear subsystem at dc. The main advantages of this technique include its simplicity and the low testing time. Two examples are given to illustrate the effectiveness of the method and its robustness in the presence of noise and input signal distortion. Transfer functions are then fitted to the nonparametric frequency responses using techniques considered in [5]. The model order is first selected using several criteria such as the analysis of statistical indicators, zeros and poles, and frequency domain residuals [5, 146].

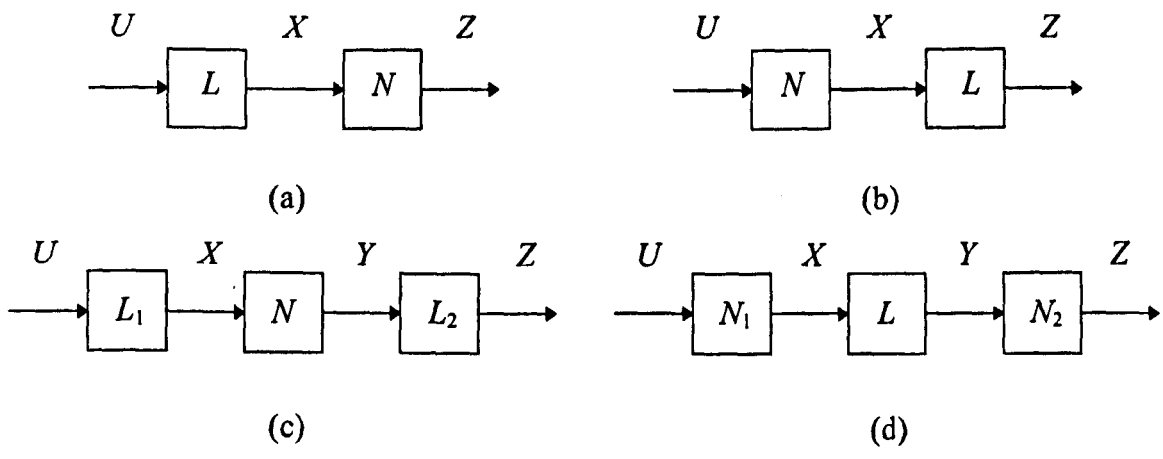

Figure 9.1. Block-oriented models: (a) Wiener, (b) Hammerstein, (c) Wiener-Hammerstein and (d) Hammerstein-Wiener. *L*: Linear dynamics; *N*: Static nonlinearity.

9.3 Volterra Kernels

Block-oriented models can be characterised using Volterra functional representation, which may be considered as a generalisation of the power series to nonlinear systems with memory [147]. For a causal, stable and time-invariant system, the time domain output $y(t)$ of a Volterra nonlinear system is given by

$$y(t)=\int_0^t h_1(\tau_1)u(t-\tau_1)d\tau_1+\int_0^t\int_0^t h_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)d\tau_1 d\tau_2+\ldots$$

$$+\int_0^t\int_0^t\ldots\int_0^t h_n(\tau_1,\tau_2,\ldots,\tau_n)u(t-\tau_1)u(t-\tau_2)\ldots u(t-\tau_n)d\tau_1 d\tau_2\ldots d\tau_n \qquad (9.1)$$

where $h_1(\tau_1)$, $h_2(\tau_1, \tau_2)$ and $h_n(\tau_1, \tau_2, \ldots, \tau_n)$ are the linear kernel, second order kernel and nth order kernel respectively; u is the input; and τ is a time variable with the subscript denoting the dimension. For most practical systems, the effect of higher order kernels decreases with increasing n, thus allowing a truncated series to be used [141]. An important advantage of the Volterra kernel is that it is valid for both time and frequency domain kernels. However, the latter is more often used in system identification since the complex multiple integrals of the time domain expression can be replaced by straightforward multiplications. The n-dimensional Laplace transforms of these kernels are given by

$$Y_n(s_1,s_2,\ldots,s_n)=H_n(s_1,s_2,\ldots,s_n)U(s_1)U(s_2)\ldots U(s_n) \qquad (9.2)$$

where U, Y and H are the Laplace transforms of u, y and h respectively; and s is the Laplace operator with the subscripts representing the frequencies of the particular dimensions. The Volterra kernel possesses symmetry [140, 148] which is given by

$$H_n(s_1,s_2,\ldots,s_n)_{sym}=\frac{1}{n!}\left\{H_n(s_1,s_2,\ldots,s_n)_{asym}+\ldots+H_n(s_n,s_{n-1},\ldots,s_1)_{asym}\right\} \qquad (9.3)$$

where a symmetrical kernel is obtained by making all possible combinations of the arguments of the asymmetric kernel. Only a symmetrical kernel is guaranteed to be unique for a given system [148].

The output of block-oriented models, such as the Wiener-Hammerstein model considered in this chapter, can be easily evaluated using a multi-dimensional Laplace transform. If a double sided frequency vector is used, the output of the nonlinear system can be calculated at all possible frequency combinations of the input signal. For a Wiener-Hammerstein model, and considering the second order kernels,

$$|H_2(s_1,s_2)| = |L_1(s_1)|.|L_1(s_2)|.|L_2(s_1+s_2)| \tag{9.4}$$

$$\angle H_2(s_1,s_2) = \angle L_1(s_1) + \angle L_1(s_2) + \angle L_2(s_1+s_2) \tag{9.5}$$

where L_1 and L_2 are the linear kernels of the first and second linear subsystems respectively. It is important to note that not all possible frequency combinations have to be evaluated, since the Volterra kernel is symmetrical along the $f_1 = f_2$ and $f_1 = -f_2$ diagonals [140, 141], where f_1 and f_2 are the values of the first and second dimensions of the input frequencies in hertz respectively. (The relationship between f and s is $f = \omega/2\pi = s/j2\pi$, where ω is the angular frequency in rad s^{-1}.)

9.4 Identification of Linear Subsystems using Linear Interpolation in the Frequency Domain (LIFRED)

9.4.1 Signal Design

The harmonic content of the input signal is of paramount importance as a signal which is badly designed may introduce unwanted distortions at the test frequencies and the harmonics at which the Volterra kernel is to be measured. The distortions at the test frequencies can be classified into Type I and Type II [149, 150]. Type I distortions are caused by the combination of a test frequency with a pair of equally positive and negative frequencies, resulting in a contribution falling at that particular test frequency

[149, 150]. If the nonlinearity is even order, all these contributions fall at dc. Type II distortions are caused by any other combinations which are not classified as Type I, which result in a contribution at a test frequency [149, 150]. It is possible to design signals which do not suffer from any Type II distortions, for a given order of nonlinearity, by proper selection of their harmonic components. These signals are termed *no interharmonic distortion* (NID) multisines, and they possess a sparse spectrum with a log-tone appearance [149]. Additionally, it is equally important that all contributions at the estimation lines in the output are due to only one combination of the input frequencies in order to be separable.

A procedure given in [140] allows signals with the above properties to be efficiently designed, without the time-consuming exhaustive search. An example of such a signal with 10 harmonics, given in [140], is

$$fv = [3 \quad 13 \quad 25 \quad 43 \quad 57 \quad 77 \quad 119 \quad 155 \quad 203 \quad 227] \tag{9.6}$$

which consists of only odd harmonics. This has a further advantage that measurements of the second order kernel will not be corrupted by third order nonlinearities. Higher order nonlinearities may be neglected in most practical cases since their effects decrease with increasing order [141]. If not, fourth and higher even order nonlinearities will interfere with the measurements of the second order kernel since these nonlinearities will contribute to terms at all the estimation lines. It should be noted that it is impossible to avoid this by signal design since the combination of k frequencies (k even), with (k-2)/2 pairs of them being equal but of opposite sign, will result in a contribution at the sum of the other two remaining frequencies. Furthermore, the number of contributions at each of the estimation lines will be too large and hence not separable even with the use of extensive computations. In practice, the inability of the proposed technique to deal with this is a minor disadvantage since in most cases, fourth and higher order nonlinearities are small enough to be neglected [141].

The signal consisting of the harmonics in (9.6) will be used in the examples illustrated in Section 9.6. For this signal, the kernel coverage plot is shown in Figure 9.2. Note that

the spectrum has a log-tone appearance and is relatively sparse as expected of NID multisines. There are gaps in the measurements corresponding to $f_1 = -f_2$ contributions as these fall on dc and therefore cannot be measured. If the lowest harmonic number is increased, the harmonic content of the signal can be made more uniform. However, since the interpolation is carried out in the linear kernel of the second linear subsystem (as will be discussed in Section 9.4.2), instead of in the second order kernel, the uniformity of the input harmonics would cause the estimation lines to be less uniform. In addition, increasing the lowest harmonic number in the input means that the testing time will be longer since the sampling frequency will have to be decreased in order to lower the actual frequency of this harmonic.

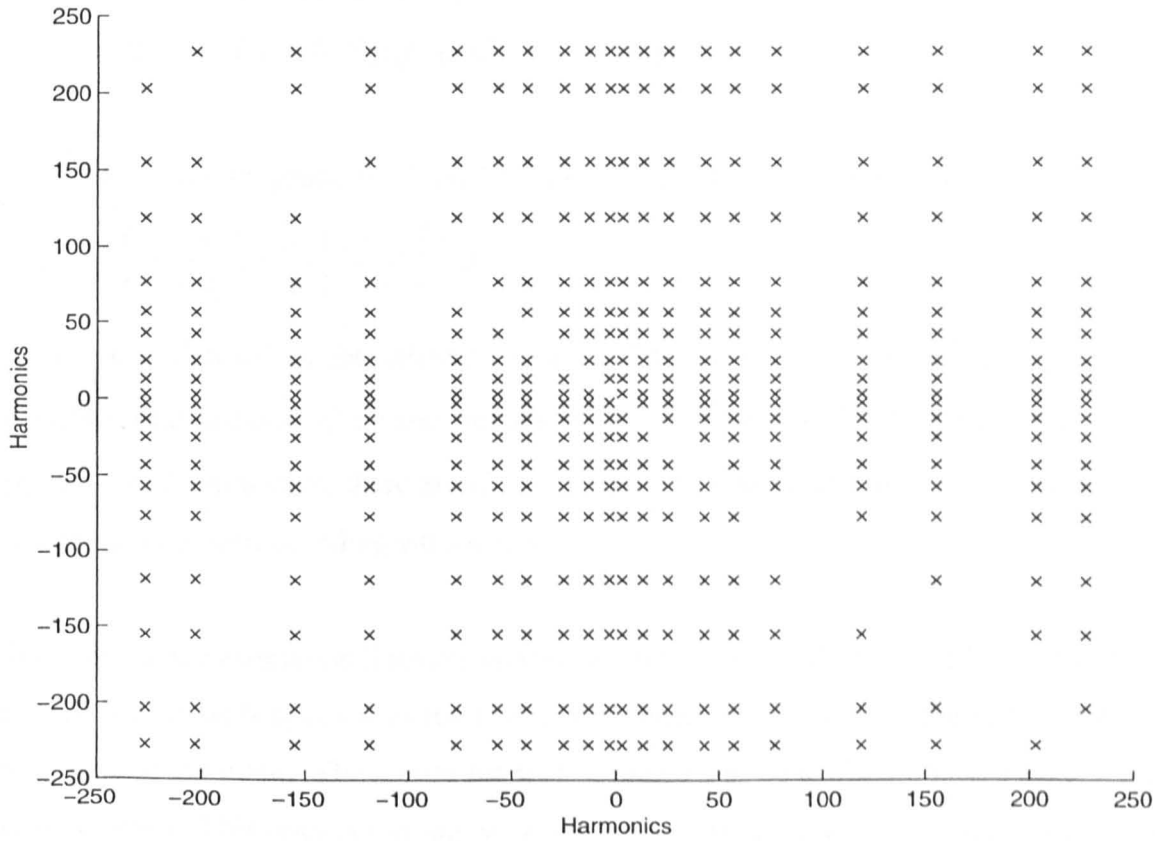

Figure 9.2. Kernel coverage plot for signal (9.6).

A further requirement in the signal is that it must sufficiently excite the linear subsystems in order that the power at the estimation lines is large enough to negate the effects of noise. In the subsequent sections, it is assumed that the approximate bandwidths of the linear subsystems are known *a priori*.

9.4.2 Calculation of the Frequency Response Gain of the Second Linear Subsystem

The second order Volterra kernel of a simple Wiener model is known to possess identical symmetry over the $f_1 = f_2$ and $f_1 = -f_2$ diagonals [148]. For a Wiener-Hammerstein model, there is still symmetry in both diagonals, but the symmetry along $f_1 = f_2$ is not identical to that along $f_1 = -f_2$. Applying this to the model in Figure 9.1(c), it is possible to extract the four-fold symmetry in Z and to attribute this to the Wiener subsystem (from U to Y). The remaining part with two-fold symmetry can then be attributed to the contribution of the linear subsystem L_2. This can be formulated as

$$|Z(s_1 + s_2)| = |L_1(s_1)|.|L_1(s_2)|.|L_2(s_1 + s_2)|.|U(s_1)|.|U(s_2)| \tag{9.7A}$$

$$|Z(s_1 - s_2)| = |L_1(s_1)|.|L_1(-s_2)|.|L_2(s_1 - s_2)|.|U(s_1)|.|U(-s_2)| \tag{9.7B}$$

From the symmetry property, $|L_1(s_2)| = |L_1(-s_2)|$ and $|U(s_2)| = |U(-s_2)|$. Hence,

$$\frac{|Z(s_1 + s_2)|}{|Z(s_1 - s_2)|} = \frac{|L_2(s_1 + s_2)|}{|L_2(s_1 - s_2)|} = \alpha \tag{9.8}$$

where α is defined as the attenuation ratio. The values of α are calculated for all possible combinations of the test frequencies f_1 and f_2, where $f_1 \neq f_2$. Thus for an input signal with F harmonics, there are $(F^2 - F)/2$ pairs of harmonics in the estimation lines linked by their corresponding values of α.

The gains at the estimation lines are now ready to be calculated using LIFRED. The first pair is chosen with s_2 equal to the lowest harmonic and s_1 equal to the second lowest harmonic in the input. (The reason for such a choice will be explained later.) $|L_2(s_1 - s_2)|$ is set to unity. This does not in any way impose a restriction on the technique since the overall gain in the system can be equally well divided between the subsystems L_1 and L_2. $|L_2(s_1 + s_2)|$ is calculated using (9.8). The rest of the pairs are now fitted in a sequence such that the harmonic s_2 is increased with every increasing s_1. Starting from the first pair in the sequence which has not been fitted, if the harmonic $s_1 - s_2$ does not lie in between at least two points which have already been estimated, the pair $|L_2(s_1 - s_2)|$ and $|L_2(s_1 + s_2)|$ is skipped over and the next pair is considered, and so on until another

pair has been successfully fitted. The program will then return to the pair which it previously failed to fit. The procedure is repeated until the gains of all the points have been calculated. In each case, the first point in the pair (the one with the lower frequency) is calculated using linear interpolation between the closest existing points on its left and right, and the second point is evaluated using (9.8). This can be summarised below, where *fv* is the vector representing the input harmonics; $\alpha_{i,j}$ denotes the attenuation ratio between $|L_2(fv(i) + fv(j))|$ and $|L_2(fv(i) - fv(j))|$; *minus* is the vector of the difference in input harmonics, arranged in an increasing order of s_1 and s_2; and *breakable* is a variable governing the flow of the program by allowing it to break from a certain point when necessary.

```
k = 1
for i = 2 to F
        for j = 1 to (i-1)
                calculate α_{i,j} from (9.8)
                minus(k) = fv(i) - fv(j); k = k + 1
        end
end
|L2(fv(2) - fv(1))| = 1; |L2(fv(2) + fv(1))| = α_{2,1}; min_k = 2
repeat until all points have been fitted
        breakable = 0
        for k = min_k to (F^2 - F)/2
                if minus(min_k) lies between at least two points which have been fitted
                        |L2(fv(i) - fv(j))| is calculated using linear interpolation
                        |L2(fv(i) + fv(j))| = α_{i,j} * |L2(fv(i) - fv(j))|
                        if breakable == 1
                                break
                        end
                else
                        breakable = 1; min_k = k
                end
        end
end
```

The reason behind the choice of frequencies for the first harmonic pair is that since the input signal has a sparse spectrum with log-tone appearance, the first pair will have the closest distance between the harmonics in the pair (due to the smallest value possible for s_2). It is then much safer to assume linearity in the gain between these harmonics. The linearity between points close together is the fundamental concept in the LIFRED technique. An additional advantage of starting with the lower frequencies is that most systems have a higher gain at low frequencies compared to that at high frequencies. Thus, the measurements at low frequencies have a higher accuracy and should be calculated first in order not to introduce a bias or a scatter of points into the frequency response estimates.

9.4.3 Calculation of the Frequency Response Gain of the First Linear Subsystem

When $s_1 = s_2 = s$, (9.7A) can be simplified to

$$|Z(2s)| = |L_1(s)|^2 . |L_2(2s)| . |U(s)|^2 \tag{9.9}$$

The values of $|L_2(2s)|$ are calculated by linear interpolation (or linear extrapolation for the lowest and highest harmonics) on the frequency response gain curve already obtained using the procedure given in Section 9.4.2. Then,

$$|L_1(s)| = \frac{1}{|U(s)|} \cdot \sqrt{\frac{|Z(2s)|}{|L_2(2s)|}} \tag{9.10}$$

The program flow is shown below :

```
for i = 1 to F
        calculate |L2(2fv(i))| using linear interpolation (or extrapolation for i = 1 and F)
        calculate |L1(fv(i))| from (9.10)
end
```

9.4.4 Simultaneous Calculation of the Frequency Response Phases of the First and Second Linear Subsystems

The symmetry that exists in the gain response unfortunately cannot be similarly utilised in the phase response. It is thus more complicated to estimate the phases of the linear subsystems compared with their gain responses. The phase of the output is given by the sum of the phases of the input and the second order kernel given in (9.5).

$$\angle Z(s_1+s_2)=\angle L_1(s_1)+\angle L_1(s_2)+\angle L_2(s_1+s_2)+\angle U(s_1)+\angle U(s_2) \qquad (9.11A)$$

$$\angle Z(s_1-s_2)=\angle L_1(s_1)-\angle L_1(s_2)+\angle L_2(s_1-s_2)+\angle U(s_1)-\angle U(s_2) \qquad (9.11B)$$

Subtracting (9.11B) from (9.11A),

$$\begin{aligned}\angle L_2(s_1+s_2)-\angle L_2(s_1-s_2) &= \angle Z(s_1+s_2)-\angle Z(s_1-s_2)-2\angle L_1(s_2)-\angle 2U(s_2)\\ &=\gamma \qquad (9.12)\end{aligned}$$

where γ is defined as the difference term.

The first pair of harmonics to be evaluated is the pair with s_1 and s_2 equal to the fourth and third lowest harmonics in the input respectively. (The reason for this choice will become clear later.) The term $\angle L_1(s_2)$ in (9.12) can be assigned an arbitrary value ϕ since the overall delay in the Wiener-Hammerstein model can be equally well attributed to the linear subsystems L_1 and L_2. This step does not violate the generality of the results obtained. However, from (9.12), the values of $\angle L_2(s_1 + s_2)$ and $\angle L_2(s_1 - s_2)$ cannot be identified individually. Hence another equation is needed to solve these uniquely. From (9.11A), by replacing s_1 with s_2,

$$\angle L_2(2s_2)=\angle Z(2s_2)-2\angle L_1(s_2)-\angle 2U(s_2) \qquad (9.13)$$

With $\angle L_2(2s_2)$ and γ calculated from (9.13) and (9.12) respectively, the values of $\angle L_2(s_1 + s_2)$ and $\angle L_2(s_1 - s_2)$ can be calculated by assuming a straight line joining the points $\angle L_2(s_1 + s_2)$, $\angle L_2(s_1 - s_2)$ and $\angle L_2(2s_2)$. $\angle L_1(s_1)$ can then be calculated using (9.11A). The subsequent pairs are taken with s_1 increasing from the fourth lowest harmonic to the highest harmonic in the input signal, and s_2 kept constant. For these pairs of harmonics, the first point (the one with the lower frequency) is calculated using

linear interpolation between the closest points on its left and right. The second point in the pair is then evaluated using (9.12). The phases obtained are unwrapped as necessary to produce a smooth curve. At the end of this procedure, the phases of L_1 would have been estimated from the third lowest harmonic to the highest harmonic using (9.11A). The phases at the first and second lowest harmonics are fitted through linear interpolation using the phase of the third lowest harmonic and the phase at dc, which is assumed to be known *a priori*.

With a complete set of phases estimated for L_1, the phases of L_2 at the remaining harmonics can be evaluated. First, the pairs (s_1, s_2) equal to (third lowest harmonic, second lowest harmonic) and (second lowest harmonic, lowest harmonic) are fitted with the higher frequency in each pair calculated using linear interpolation and the lower frequency calculated using (9.12). This avoids the need for extrapolation, which is less accurate. Next, s_2 is set to increase from the lowest to the second highest harmonic in the input, with the third lowest harmonic being skipped over. For each value of s_2, s_1 is set to increase from $s_2 + 1$ to the highest harmonic in the input. If the points in the (s_1, s_2) pair have not yet been evaluated, the first points in the pairs are calculated using linear interpolation and the second points are calculated from (9.12). The phases are unwrapped as necessary to form a smooth curve. The procedure can be summarised on the next page, where $\gamma_{i,j}$ denotes $\angle L_2(fv(i) + fv(j)) - \angle L_2(fv(i) - fv(j))$.

The reasons behind the choice of the first pair of harmonics will now be explained. The first value of s_2 is a compromise between several criteria. A small value of s_2 means that the pairs of harmonics formed using the constant value of s_2 will have points in each pair close to one another, but far from those of the other pairs. Very likely, the frequency range of a pair does not overlap with those of the other pairs, making linear interpolation extremely difficult. On the other hand, if s_2 is large, the first few points fitted will be very far from one another, and the assumption of linearity between the points will become questionable. Also, the phases of L_1 have to be linearly interpolated between this value of s_2 and dc. With a smaller value of s_2, greater confidence can be placed in doing this. Hence, the choice of s_2 as the third lowest harmonic in the input signal is a compromise between conflicting requirements.

$\angle L_1(fv(3)) = \phi$

for $i = 4$ to F

calculate $\gamma_{i,3}$ from (9.12)

if $i = 4$

calculate $\angle L_2(2fv(3))$ from (9.13)

$\angle L_2(fv(i) + fv(3))$ and $\angle L_2(fv(i) - fv(3))$ is fitted assuming a straight line between $\angle L_2(fv(i) + fv(3))$, $\angle L_2(fv(i) - fv(3))$ and $\angle L_2(2fv(3))$

else

$\angle L_2(fv(i) - fv(3))$ is fitted using linear interpolation

$\angle L_2(fv(i) + fv(3)) = \angle L_2(fv(i) - fv(3)) + \gamma_{i,3}$

end

calculate $\angle L_1(fv(i))$ from (9.11A)

$\angle L_2(fv(i) + fv(3))$, $\angle L_2(fv(i) - fv(3))$ and $\angle L_1(fv(i))$ are unwrapped as necessary

end

calculate $\angle L_1(fv(1))$ and $\angle L_1(fv(2))$ using linear interpolation

for $i = 3, j = 2$ and then $i = 2, j = 1$

calculate $\gamma_{i,j}$ from (9.12)

$\angle L_2(fv(i) + fv(j))$ is fitted using linear interpolation

$\angle L_2(fv(i) - fv(j)) = \angle L_2(fv(i) + fv(j)) - \gamma_{i,j}$

$\angle L_2(fv(i) + fv(j))$ and $\angle L_2(fv(i) - fv(j))$ are unwrapped as necessary

end

for $j = 1$ to $(F\text{-}1)$ and $j \neq 3$

for $i = (j+1)$ to F

if $(i \neq 3, j \neq 2)$ and $(i \neq 2, j \neq 1)$

calculate $\gamma_{i,j}$ from (9.12)

$\angle L_2(fv(i) - fv(j))$ is fitted using linear interpolation

$\angle L_2(fv(i) + fv(j)) = \angle L_2(fv(i) - fv(j)) + \gamma_{i,j}$

$\angle L_2(fv(i) + fv(j))$ and $\angle L_2(fv(i) - fv(j))$ are unwrapped as necessary

end

end

end

The choice of the first value of s_1 is less complicated. Since s_1 could not be smaller than s_2 in order to apply (9.12), it is logical to start with s_1 equal to the next harmonic in the signal from s_2. This has a further advantage since with a smaller s_1, the probability of the frequency range of the next pair overlapping that of the first pair is higher, bearing in mind the log-tone appearance of the input multisine.

It is important to note that the sequence of the harmonic pairs to be evaluated (for both gain and phase) is dependent on the actual harmonic content of the input signal. Thus, slight modifications to this sequence may be necessary if the input signal used is different from that in (9.6). However, the concepts suggested in this chapter (the separation of the dynamics of the two linear subsystems and the estimation using linear interpolation) will also work with other NID multisine signals.

9.4.5 Advantages of the LIFRED Technique

The LIFRED technique offers many advantages over existing methods for identifying second order Volterra kernels. Firstly, the testing time required is drastically shortened, since only a single experiment is needed. In contrast, the technique given in [142] requires multiple small-signal sine experiments for each large-signal multisine. (The use of two large-signal multisines were suggested in the paper.) Also, in [14], the identification of the Wiener-Hammerstein model simulated on an electronic circuit was carried out using 423 separate experiments.

Secondly, the computational burden needed for LIFRED is much less compared with algorithms requiring optimisations or recursive iterations [14, 142 - 144]. This, together with the short testing time, favours the use of LIFRED in autotuning applications where the controller parameters are updated at short time intervals.

A further advantage of the technique is the simplicity of the algorithm since all the steps are transparent to the user. The estimates of the frequency response can be easily calculated since neither optimisation nor iterative algorithms are used. This allows the

possibility of manually checking the results obtained and any problems can be corrected in a similar manner.

9.5 Parametric Estimation and Model Selection

9.5.1 Parametric Estimation using *elis*

The transfer functions of L_1 and L_2 in Figure 9.1(c) can be estimated using *elis* in the MATLAB Frequency Domain System Identification Toolbox [5]. The frequency responses at X and Y are obtained as $U.L_1$ and Z/L_2 respectively. Alternatively, the estimated frequency responses are used as the output data, and the input data is set to unity for all frequencies, as was done in [141]. If the pure time delay of the system is known, then it should be fixed at this value in the simulation. If the variances of the input and output Fourier coefficients are available, these should also be specified before the start of the simulation. However, when no averaging has been carried out, the variance data is unavailable and should be left at the default argument values of unity, unless there is any other information present which indicates otherwise. The estimation is done in the z-domain as the simulation is conducted in discrete time, using a digital signal.

9.5.2 Model Selection

Model selection forms an important part of the identification process. It is essential to avoid the situations of undermodelling, where the model order is too low, and overmodelling, where the model order is too high. When undermodelling occurs, the model is not able to capture some of the important characteristics of the actual system. In the case of overmodelling, the model has redundant parameters and would normally require longer computational times and a larger number of iterations for convergence. While these situations should be avoided, it is necessary to bear in mind that a linear transfer function is often an approximate model of a physical system, which cannot be modelled accurately with a limited number of parameters. Some effects of nonlinearities or distributed parameters may be present, which are not adequately modelled by the

selected model. Thus a limited modelling error is to be expected. This in turn complicates the process of model selection. Fortunately, there are techniques available to detect the situations of undermodelling and overmodelling and these are treated in [5, 146, 151].

9.5.2.1 Statistical Indicators

When a model is bad, the best indicator is the large value of the cost function K [5]. In the case of a good model, $2K$ is a random variable with a χ^2-distribution and $2F$ - n_p degrees of freedom, where F is the number of estimation lines and n_p is the number of free parameters. Consequently, if no modelling errors are present, the cost function should reach a theoretical minimum [152, 153] of

$$K_{\min} = F - \frac{n_p}{2} \tag{9.14}$$

Thus a comparison of the actual and the theoretical minimum of the cost function allows an evaluation of the goodness of a particular model. Additional advantages of using cost functions are that these estimates possess the properties of being asymptotically unbiased, efficient and robust to various time domain disturbances. However, special care has to be taken to use correct variance values since these will have a scaling effect on the cost function.

A second indicator is the mean model error h_{mean} [5], which shows the extent of modelling errors present. This should be interpreted in conjunction with the mean absolute value of the transfer function H_{mean}, since with a larger value of H_{mean}, the value of h_{mean} is also likely to increase. In addition to this, incorrect variance values may lead to an imaginary value of h_{mean} (if the variance values are too large) or a large value of h_{mean} (if the variance values are too small).

A further possibility is to calculate Akaike's information criterion (AIC) [146, 151, 154 - 156] when searching for the best order model. This is given by

$$\mathrm{AIC} = 2(K + n_p) \tag{9.15}$$

where K is the cost function. The model with the lowest AIC value is to be preferred.

All the above indicators are available as outputs when running *elis*.

9.5.2.2 Analysis of Zeros and Poles

The analysis of zeros and poles can provide useful information about the quality of a model. When overmodelling occurs, the redundant zeros and poles will have large uncertainties. The plots of the zeros and poles with their confidence ellipses can be generated using the function *plotelpz* while their standard deviations can be assessed using *stdpz*, both which are available in the Frequency Domain System Identification Toolbox. Additionally, unnecessary zeros and poles often appear in pairs, almost cancelling one another. They will have a high correlation and this can also be detected using *stdpz*.

9.5.2.3 Analysis of Residuals

The frequency domain residuals between the measured and the model frequency responses are calculated. Any model errors are a result of either stochastic effects, undermodelling or nonlinear distortion. A technique to distinguish between these was proposed in [157] which involves the computation of the normalised autocorrelation of the residuals given by

$$R_{ee}(m) = \frac{1}{F-m}\sum_{k=1}^{F-m}\frac{(H_m(\mathrm{j}\omega_k) - H(\mathrm{j}\omega_k, \hat{P}))(H_m(\mathrm{j}\omega_k) - H(\mathrm{j}\omega_k, \hat{P}))^*}{\sigma^2(\mathrm{j}\omega_k)} - \delta(m-0) \tag{9.16}$$

where $H_m(\mathrm{j}\omega_k)$ and $H_m(\mathrm{j}\omega_k, \hat{P})$ are the measured and estimated frequency response functions respectively; $\sigma^2(\mathrm{j}\omega_k)$ is the standard deviation of $H_m(\mathrm{j}\omega_k)$; and δ is the Kronecker delta function. m represents the mth harmonic in the harmonic vector while $*$ denotes complex conjugate.

If $|R_{ee}|$ is small, the model is good. However, if it is large, with a sharp peak at zero lag, the residuals are uncorrelated (except at zero lag) and the errors are due to either stochastic effects or nonlinear distortion. At this point, the value of the cost function should be taken into account. If this is small, the errors are due to stochastic effects. Otherwise, the errors are a result of unmodelled nonlinear distortion. If $|R_{ee}|$ is large and broad, this suggests undermodelling whereby the difference between the measured and estimated frequency responses vary smoothly, and are correlated. Such an analysis can be performed easily using the function *rdueelis* (also in the Frequency Domain System Identification Toolbox).

9.6 Simulation Examples

Two simulation examples are presented in this section. In the first example, both the linear subsystems L_1 and L_2 have frequency responses which can be reasonably well represented using straight lines, especially in the gain responses. This is not so in the second example in which L_1 and L_2 were intentionally chosen such as to allow a direct comparison between the effectiveness of using LIFRED under some sort of 'best case' and 'worst case' scenarios.

9.6.1 Example 1

In this example, L_1 is a second order system with time constants of 1 second and 10 seconds. The gain falls to half its maximum at $\omega = 0.170$ rad s^{-1}, which corresponds to 0.0271 Hz. The linear subsystem L_2 is a first order process with a time constant of 5 seconds and the gain is half that of its maximum at $\omega = 0.346$ rad s^{-1}, or a cyclic frequency of 0.0551 Hz. The input multisine is of length $N = 4096$ and has the harmonics given in (9.6). Using a sampling frequency of 1 Hz, the highest estimation lines used in estimating L_1 and L_2 are at 0.0554 Hz and 0.111 Hz respectively thus covering approximately twice the frequency ranges mentioned above.

The second order Volterra kernels at Y and Z (see Figure 9.1(c)) are plotted in Figures 9.3 and 9.4 respectively. Note that there is a four-fold symmetry in Figure 9.3 while the symmetry is two-fold in Figure 9.4.

The experiment was repeated with band-limited white noise added to the system output. The noise was first filtered using a third order Chebyshev lowpass filter with 3dB of ripple in the passband and a cut-off frequency of 0.04 Hz. (The filter coefficients were obtained using the function *cheby1* in the MATLAB Signal Processing Toolbox [158].) A signal power to noise power ratio of approximately 30dB was achieved in the output at the frequencies of interest.

A third experiment was then carried out with the input signal being distorted by the presence of nonlinearities. This may well happen in practice if the part preceding the Wiener-Hammerstein system is analogue. In [140], an example is illustrated using a servo motor system, where the ratio of the signal amplitude to the distortion amplitude obtained is approximately 60dB. In Example 1 of this chapter, the distortion amplitude was also set at 60dB below the signal amplitude.

The actual and estimated frequency responses of L_1 and L_2 are shown in Figures 9.5 and 9.6 respectively. Since the input multisine contains 10 harmonics, there are 10 frequencies used in the estimation of L_1 but 10^2 - 10 = 90 frequencies used in the estimation of L_2 (as discussed in Section 9.4.2). From these figures, it can be seen that the estimates obtained under the no noise and no input distortion condition have a very high accuracy. For the other two cases investigated, the estimates are still reasonably accurate, except that there is a bias at high frequencies in the phase response of L_2. The reason is that there was very little signal power at the output in these frequencies due to the lowpass characteristics of L_1 and L_2, hence resulting in lower accuracy of the measurements.

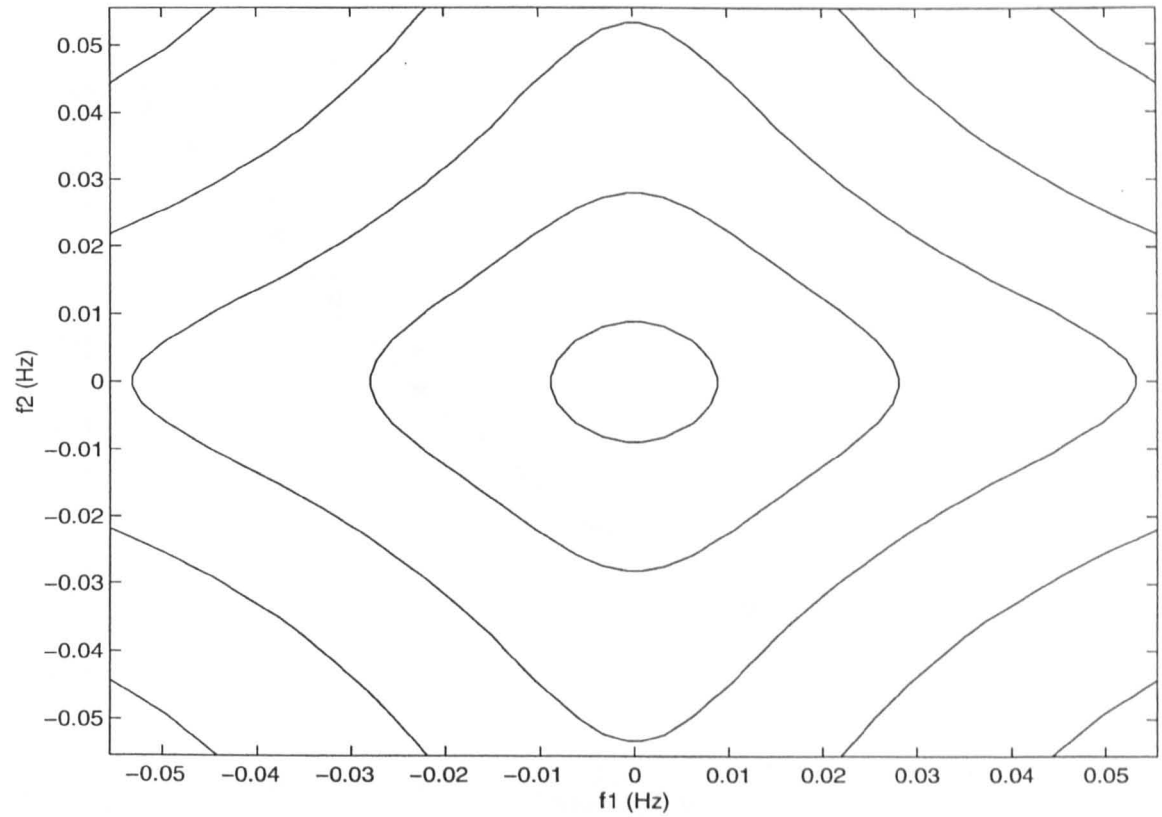

Figure 9.3. Example 1: Second order Volterra kernel at Y.

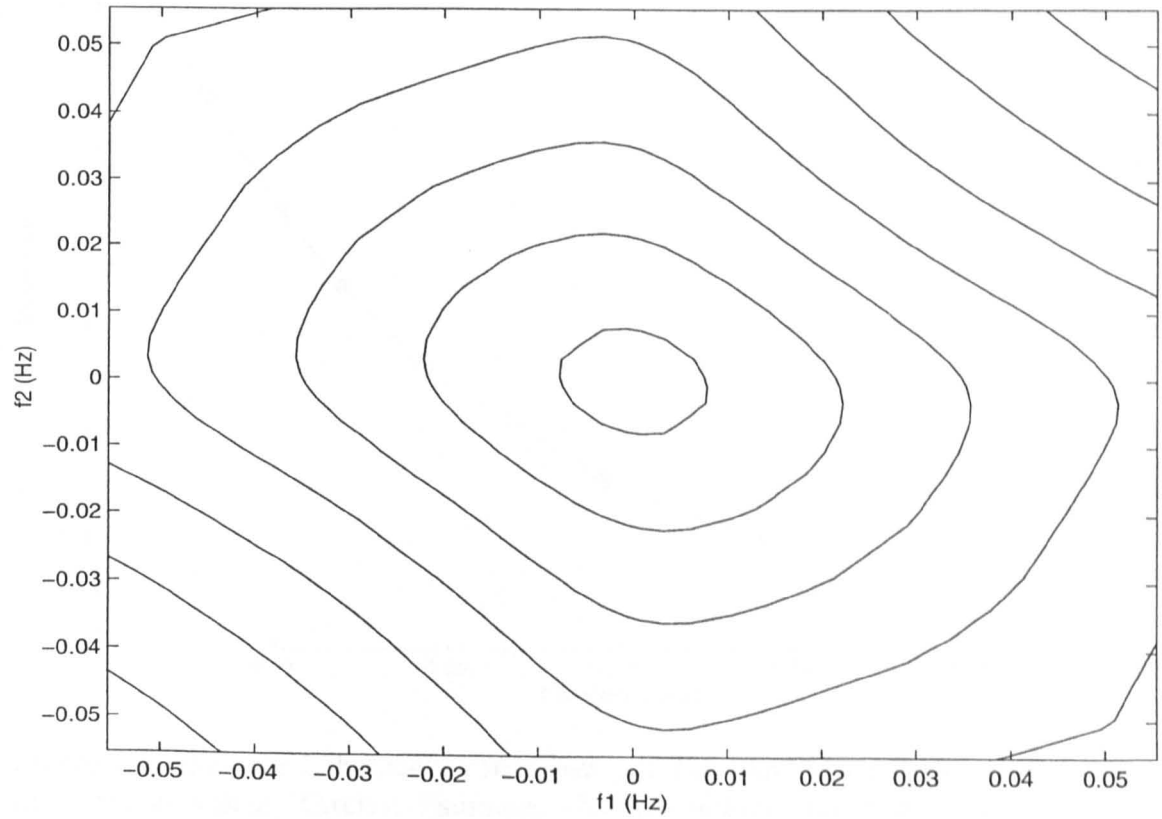

Figure 9.4. Example 1: Second order Volterra kernel at Z.

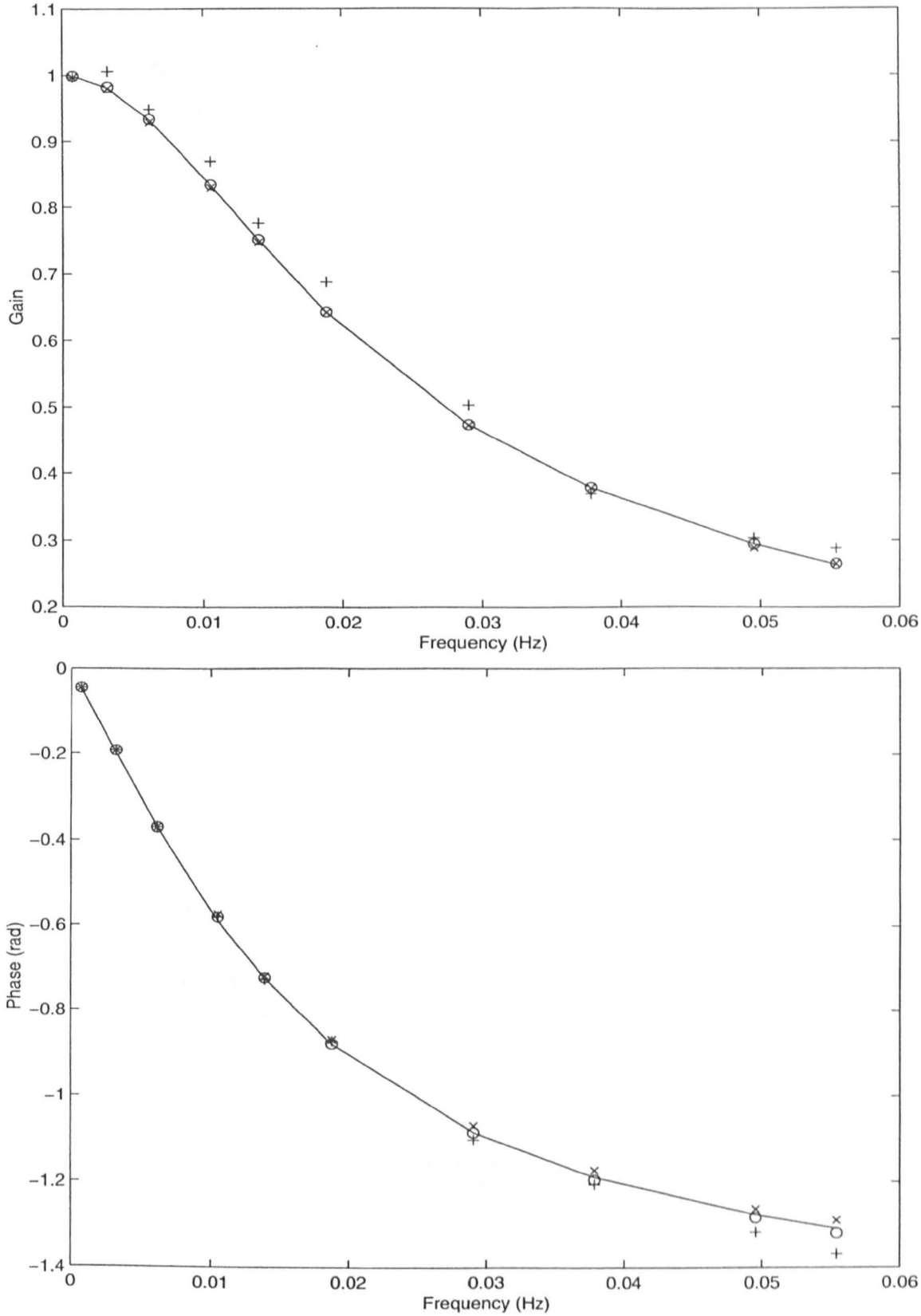

Figure 9.5. Example 1: Frequency response gain (top) and phase (bottom) of L_1. Solid line: Actual values; Circles: Estimates obtained without noise and input distortion; Plusses: Estimates obtained with noise; Crosses: Estimates obtained with input signal distortion.

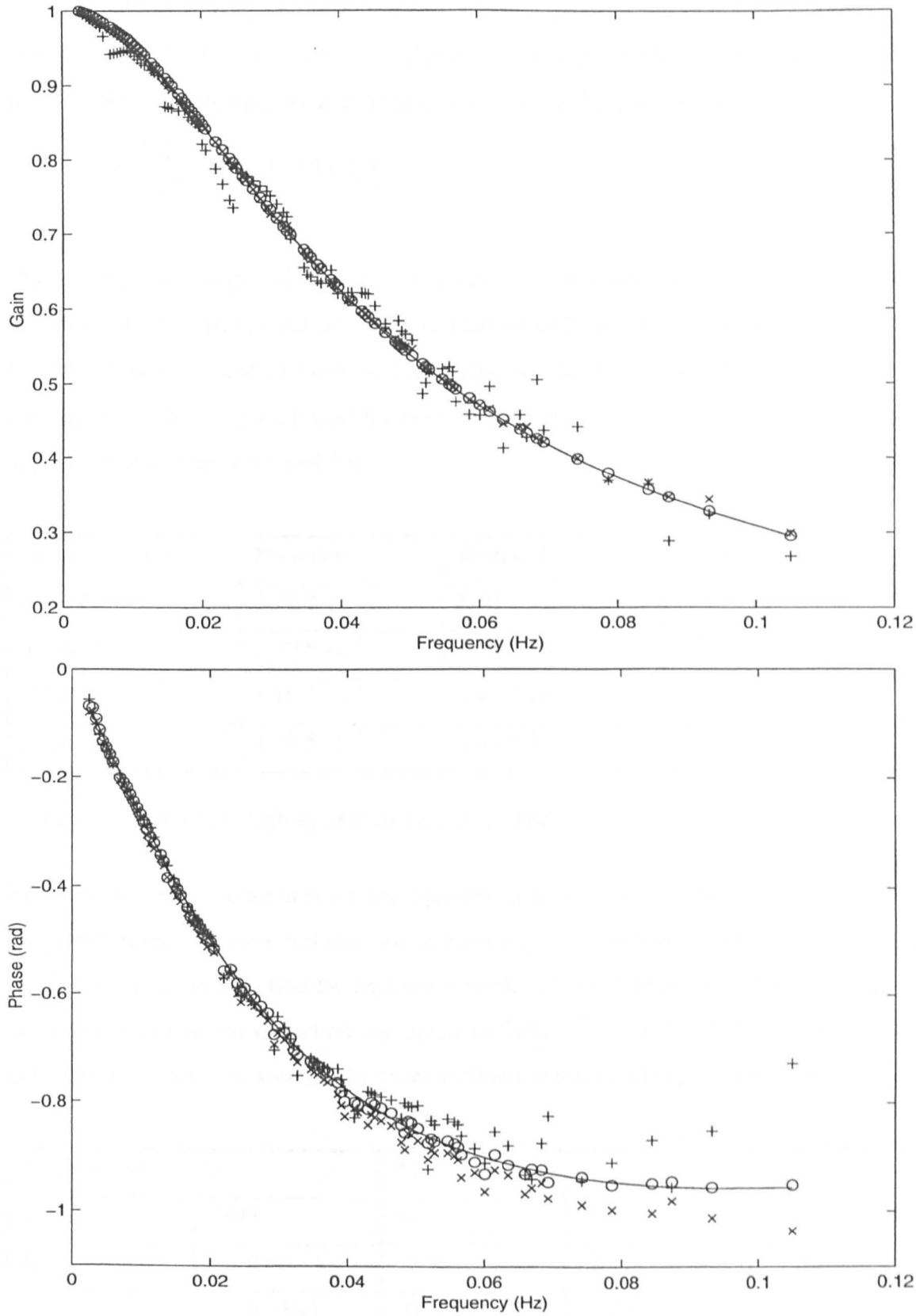

Figure 9.6. Example 1: Frequency response gain (top) and phase (bottom) of L_2. Solid line: Actual values; Circles: Estimates obtained without noise and input distortion; Plusses: Estimates obtained with noise; Crosses: Estimates obtained with input signal distortion.

A statistical measure of the deviation of the estimates from the true values, termed the complex error E in [159, 160], was calculated by computing the mean of the distances between the estimated points and the actual ones in the Nyquist plane.

$$E = \frac{1}{F}\sum_{k=1}^{F}\left|H(j\omega_k) - \hat{H}(j\omega_k)\right| \tag{9.17}$$

The mean of the squares of the distances E_S can also be used. Results obtained are given in Table 9.1. The errors obtained in the estimation of L_2 are larger than those obtained in the estimation of L_1 and this is to be expected from Figures 9.5 and 9.6. Also, the errors are larger for the noisy case than for the case with input signal distortion. This is again apparent from Figures 9.5 and 9.6.

Error estimates	No noise	With noise	With input distortion
E for L_1	$2.71 * 10^{-3}$	$2.40 * 10^{-2}$	$6.49 * 10^{-3}$
E_S for L_1	$1.29 * 10^{-5}$	$7.26 * 10^{-4}$	$4.97 * 10^{-5}$
E for L_2	$5.10 * 10^{-3}$	$2.49 * 10^{-2}$	$1.38 * 10^{-2}$
E_S for L_2	$4.09 * 10^{-5}$	$8.49 * 10^{-4}$	$2.33 * 10^{-4}$

Table 9.1. Example 1: Values of E and E_S for L_1 and L_2.

In order to obtain estimates of the transfer functions of L_1 and L_2, the estimated frequency responses were fed into *elis* and the statistical indicators discussed in Section 9.5.2.1 were analysed to find the best order model. Results obtained for the case without noise and input signal distortion are given in Tables 9.2 and 9.3 respectively. Similar conclusions could be drawn for the cases with noise and input signal distortion.

Model order	0/1	0/2	1/2	1/3
K	3790	38.0	36.5	23.6
K_{min}	9.00	8.50	8.00	7.50
h_{mean}	0.0300	0.0027	0.0026	0.0020
H_{mean}	0.652	0.656	0.656	0.655

Table 9.2. Example 1: Statistical indicators for different model orders for L_1.

Model order	0/1	0/2	1/2
K	73.2	72.4	66.6
K_{min}	89.0	88.5	88.0
h_{mean}	0.0063j	0.0064j	0.0074j
H_{mean}	0.716	0.716	0.717

Table 9.3. Example 1: Statistical indicators for different model orders for L_2.

From Table 9.2, the most clear cut indication of the best model order is the value of the cost function. With an order of 0/1, K is very large. When the order is increased to 0/2, K drops drastically and then only decreases relatively slightly with further increase in the model order. Thus the order 0/2 is the best order judging from the sharp decrease in the value of K. Although it could be argued that the decrease in K when the order is increased to 1/3 may be significant, this actually resulted in an unstable pole. The value of h_{mean} for order 0/1 is also much larger than that of the other orders taking into account the fact that H_{mean} is approximately the same for all the orders considered. This again verifies that 0/2 is the best order model for L_1. The AIC value was not considered since from (9.15), it is largely determined by the value of K (due to the small number of free parameters n_p) and hence follows the trend of the cost function.

From Table 9.3, it can be seen that the values of K are too small, and in fact smaller than K_{min}. Also the values of h_{mean} are imaginary. These indicate that the variance data is too large (from Section 9.5.2.1). As no noise was added in the experiment and no averaging had been carried out, the variance of the input Fourier coefficients was set to zero and the variance of the output Fourier coefficients had been left at the default value of unity. However, since the same variance data was used for all the different sets of model orders considered, the values of the statistical indicators obtained were similarly scaled and could therefore still be used to provide information on the best order model. From the table, there is a relatively large decrease in the value of K when the order is increased from 0/2 to 1/2. This may initially indicate that 1/2 is the best order. However, this again resulted in an unstable pole. Therefore, the final model order was chosen as 0/1.

Since the best model orders were already reasonably clear, no further investigation was carried out. For both the linear subsystems, the model orders chosen were equal to the actual orders simulated, thus proving the usefulness of the analysis conducted.

The parametric estimates of the frequency responses obtained for L_1 and L_2 are given in Table 9.4, in terms of the z-transfer function poles. Note that these are reasonably close to the true values for all the three cases investigated. The effects of biasing and scattering in the frequency response estimates caused by noise and input distortion were reduced by fitting a smooth curve over the points. Hence, the estimated transfer functions are not very susceptible to these unwanted effects.

Parameters	Actual	No noise	With noise	With distortion
Poles of L_1	0.368 (1.00)	0.378 (1.03)	0.450 (1.25)	0.338 (0.92)
	0.905 (10.00)	0.904 (9.91)	0.900 (9.49)	0.905 (10.00)
Pole of L_2	0.819 (5.00)	0.821 (5.07)	0.819 (5.00)	0.824 (5.17)

Table 9.4. Example 1: Estimated poles of L_1 and L_2 with the corresponding time constants in seconds given in brackets.

9.6.2 Example 2

In this example, L_1 is a fourth order Chebyshev filter with 5dB of ripple in the passband and a cut-off frequency of 0.05 Hz. L_2 is a sixth order Chebyshev bandpass filter with 3dB of ripple in the passband of 0.001 Hz to 0.1 Hz. Using the same input and sampling frequency as that in Example 1 gave sufficient excitation in the passbands of L_1 and L_2.

The second order Volterra kernels at Y and Z are plotted in Figures 9.7 and 9.8 respectively. Visual inspection of Figure 9.7 yielded the fact that L_1 consists of at least a linear term which is underdamped since the contours are closed and there are 'islands' in the plot [161]. From Figure 9.8, the contours are closed around more than one point at the main diagonal. This shows that the linear subsystem after the quadratic nonlinearity consists of at least an underdamped term [161].

As in Example 1, the experiment was repeated with band-limited white noise added to the system output. The noise power was approximately 30dB below the signal power at the frequencies of interest. However, the noise was not filtered by a lowpass filter as in the previous example since the overall gain response of the Wiener-Hammerstein model was not strictly lowpass. The noise therefore had a reasonably flat spectrum at the frequencies of interest.

Next, the experiment was carried out without any added noise but with nonlinear distortion in the input signal (as was done in Example 1). This assumes that the input signal could not be exactly realised in practice, due to nonlinearities preceding the Wiener-Hammerstein system. The distortion amplitude was again set uniformly at 60dB below the signal amplitude.

The actual and estimated frequency responses of L_1 and L_2 are shown in Figures 9.9 and 9.10 respectively. The inaccuracy in the first point of the gain plot in Figure 9.9 was caused by the fact that the gradient of the gain of L_2 is very steep at low frequencies, thus making extrapolation very difficult. (This is one of the two points in the LIFRED procedure which are estimated through linear extrapolation.) From Figure 9.10, there is a bias in the gain estimates in which the estimated values are mostly lower than the true values. The scaling factor here was set such that the first point in the plot (that corresponding to the lowest harmonic in the input signal) is at a gain of unity. However, in practice, the scaling factor is not normally known and the bias can be removed since the actual and the estimated frequency responses have the same shapes.

It is interesting to note that for the phase plot in Figure 9.10, there are a few points which lie relatively far away from (and below) the actual phase response. The bias in these points was caused by the inaccuracy in the phase estimate of the fourth lowest harmonic in Figure 9.9. This point was the first to be estimated for the phase of L_1 (excluding the third lowest harmonic which was placed arbitrarily) and hence the accuracy is expected to be much lower (since linear interpolation works much better when there are many points to interpolate between).

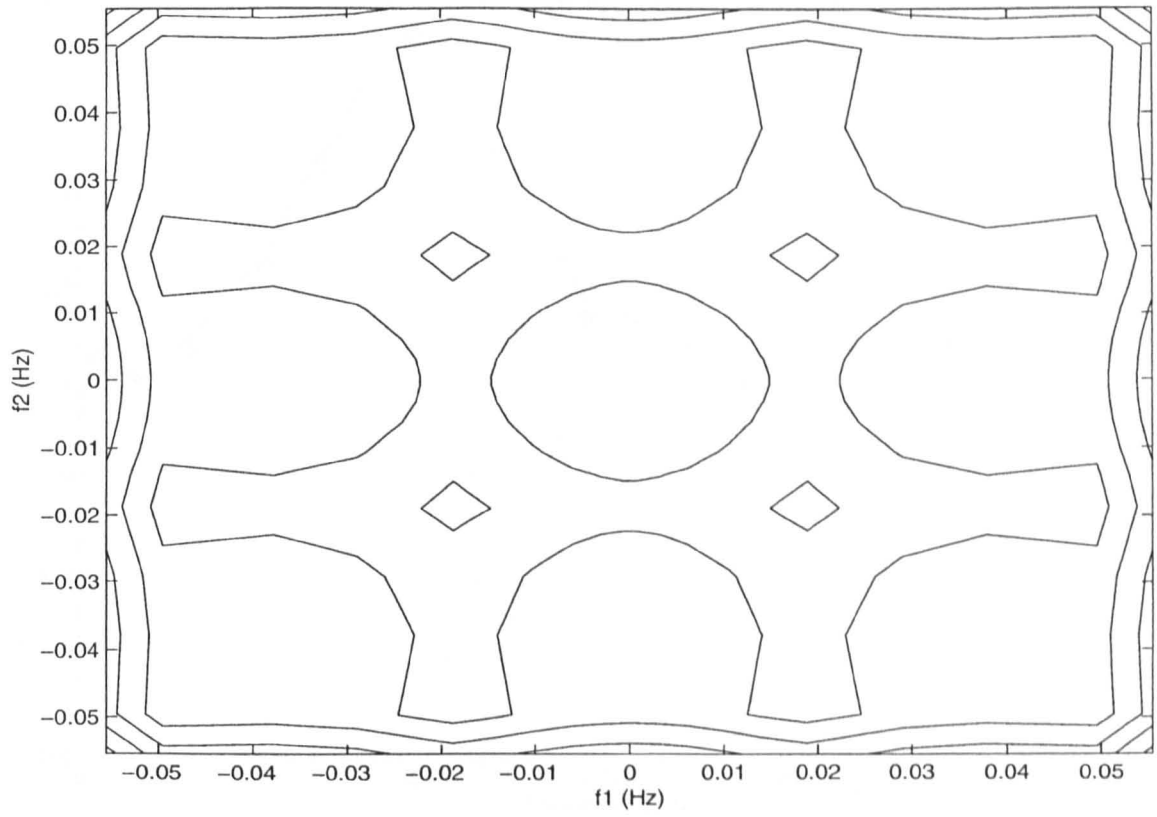

Figure 9.7. Example 2: Second order Volterra kernel at Y.

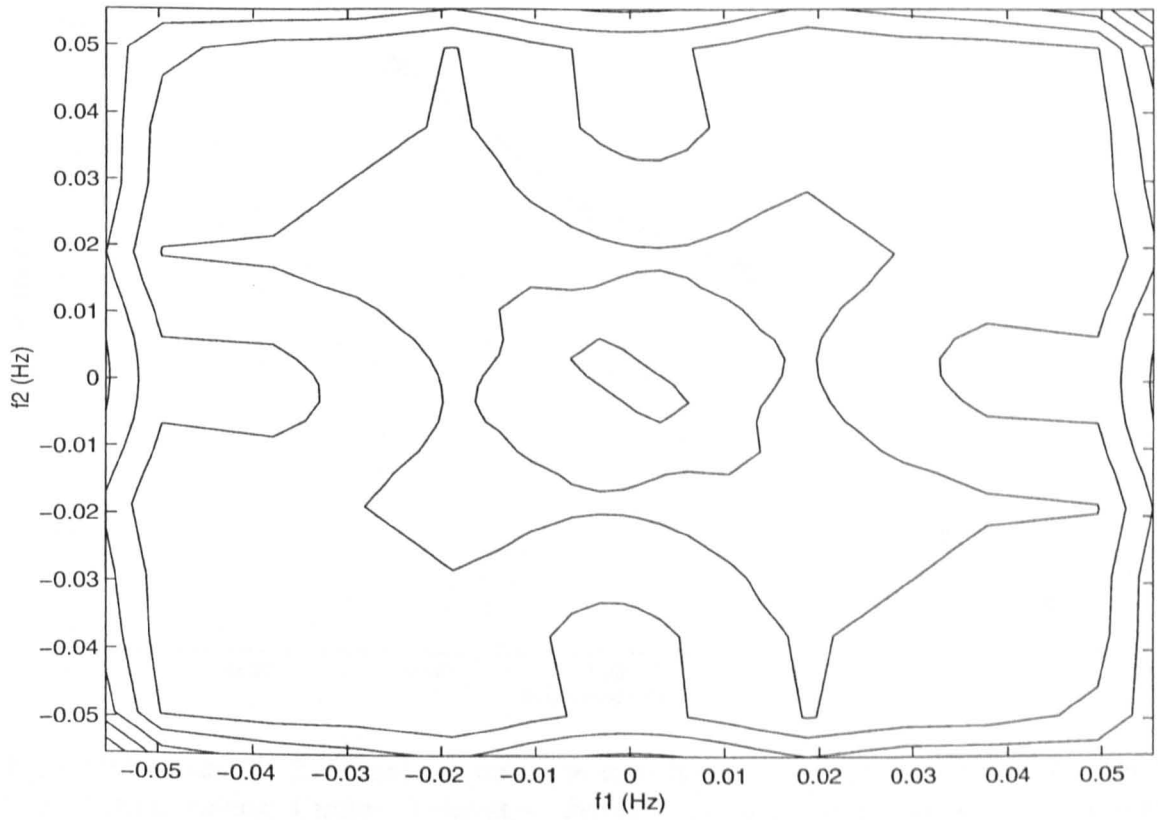

Figure 9.8. Example 2: Second order Volterra kernel at Z.

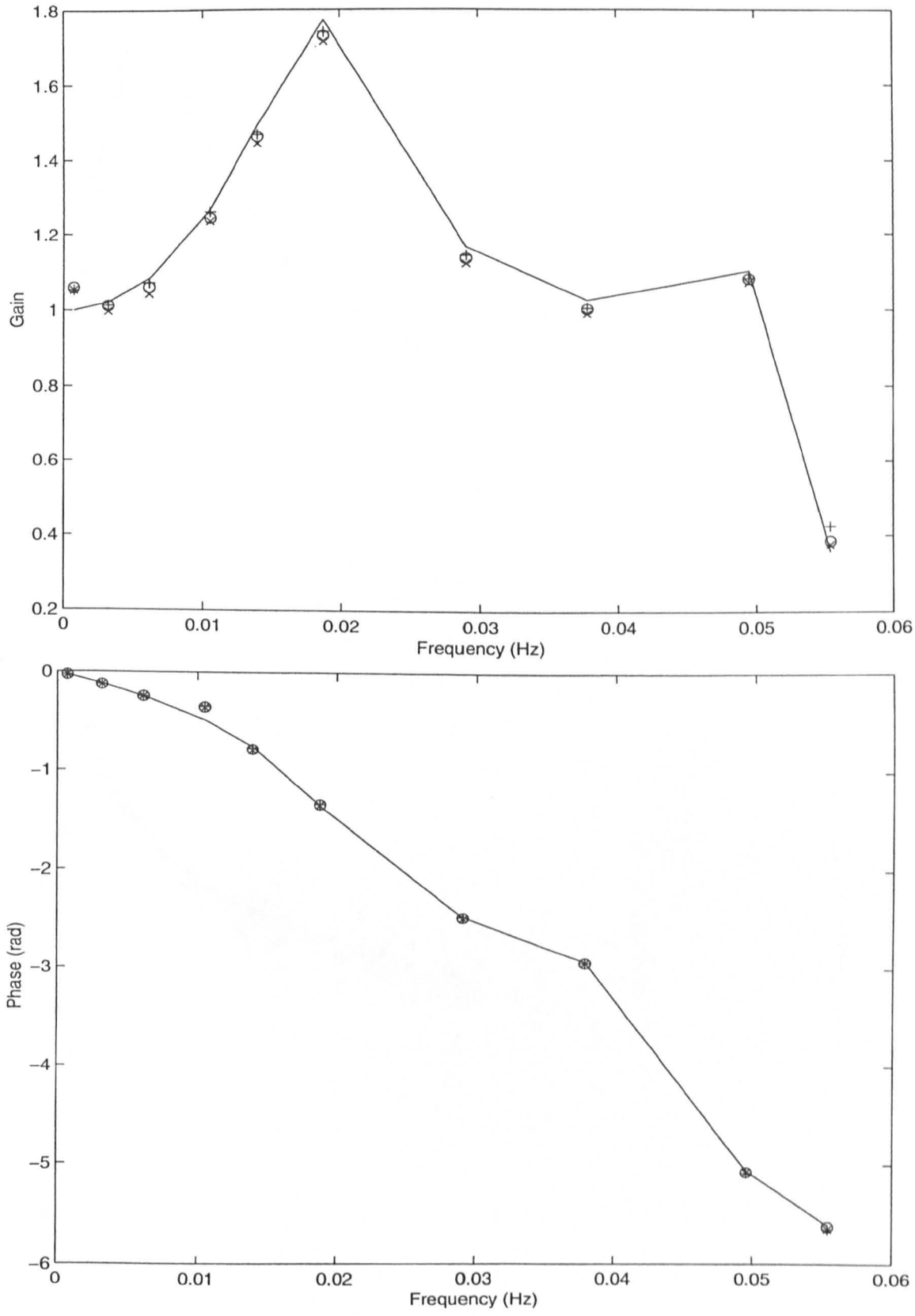

Figure 9.9. Example 2: Frequency response gain (top) and phase (bottom) of L_1. Solid line: Actual values; Circles: Estimates obtained without noise and input distortion; Plusses: Estimates obtained with noise; Crosses: Estimates obtained with input signal distortion.

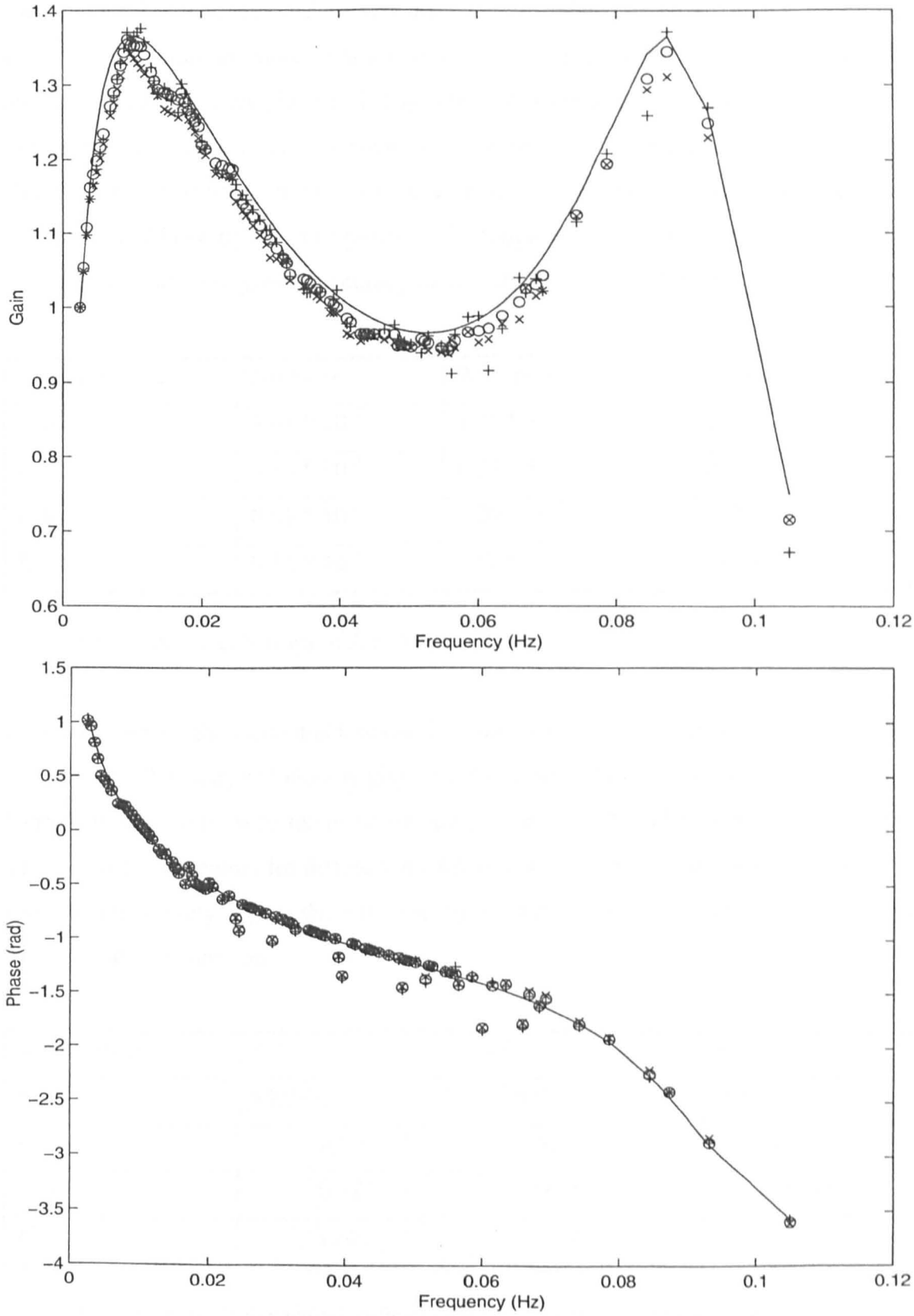

Figure 9.10. Example 2: Frequency response gain (top) and phase (bottom) of L_2. Solid line: Actual values; Circles: Estimates obtained without noise and input distortion; Plusses: Estimates obtained with noise; Crosses: Estimates obtained with input signal distortion.

The statistical values of E and E_S were again calculated and given in Table 9.5. From the table, the estimates are more accurate for L_1 and this is to be expected from Figures 9.9 and 9.10. The errors are also much larger than that obtained in Example 1. The reason behind this is that the frequency responses in Example 1 much more closely resemble a straight line (especially in the gain) than that in Example 2. Hence, the underlying assumption of linearity between points in the frequency responses is more nearly true in Example 1, leading to greater accuracy in the estimation using LIFRED.

Error estimates	No noise	With noise	With input distortion
E for L_1	$4.64 * 10^{-2}$	$4.71 * 10^{-2}$	$5.55 * 10^{-2}$
E_S for L_1	$3.84 * 10^{-3}$	$4.25 * 10^{-3}$	$4.49 * 10^{-3}$
E for L_2	$6.67 * 10^{-2}$	$7.20 * 10^{-2}$	$7.47 * 10^{-2}$
E_S for L_2	$9.54 * 10^{-3}$	$1.12 * 10^{-2}$	$9.76 * 10^{-3}$

Table 9.5. Example 2: Values of E and E_S for L_1 and L_2.

In order to obtain the estimated transfer functions for L_1 and L_2, order selection was first carried out. The analysis below is given for the case without noise and input distortion. Very similar results were obtained for the situations with added noise and distortion. The statistical indicators for different model orders for L_1 and L_2 are given in Tables 9.6 and 9.7 respectively. Again the AIC was not used due to it having an extremely similar trend as the cost function.

Model order	3/3	4/4	5/5
K	433000	12700	10600
K_{min}	6.50	5.50	4.50
h_{mean}	0.321	0.055	0.050
H_{mean}	1.08	1.12	1.12

Table 9.6. Example 2: Statistical indicators for different model orders for L_1.

Model order	2/2	4/4	6/6	8/8
K	16700	7400	7330	6750
K_{min}	87.5	85.5	83.5	81.5
h_{mean}	0.250	0.166	0.165	0.159
H_{mean}	1.10	1.12	1.12	1.12

Table 9.7. Example 2: Statistical indicators for different model orders for L_2.

From Table 9.6, K and h_{mean} drop very rapidly when the order is increased from 3/3 to 4/4. This shows that the order 4/4 is much better than 3/3. Further increasing the order does not cause significant changes in K and h_{mean}. Worse still, this gives additional pole or poles which are unstable. The final model was chosen as that of order 4/4.

Table 9.7 shows some of the orders which were considered for L_2. Again, there is a large decrease in K and h_{mean} when the order is increased from 2/2 (and 3/3 although not shown in the table) to 4/4. Any further decrease in these two indicators is relatively insignificant.

Since it was known that the actual model order was 6/6, the plots obtained using *elis* were then examined. The plots with orders 4/4 and 6/6, are illustrated in Figures 9.11 and 9.12 respectively. Comparing these, it can be seen that the model order 4/4 is the more suitable one, since there is no particular reason to have an extra peak in the gain response. From the zero-pole plot in Figure 9.12, there is a pair of complex zeros at $0.948 \pm j0.314$ and a pair of complex poles at the same position (true to three significant figures). The correlations between their real parts and their imaginary parts were 0.997 and 0.998 respectively. These were large enough for the pairs to be excluded from the final model. The standard deviations of the zeros and poles for order 4/4 were then investigated and found to be small.

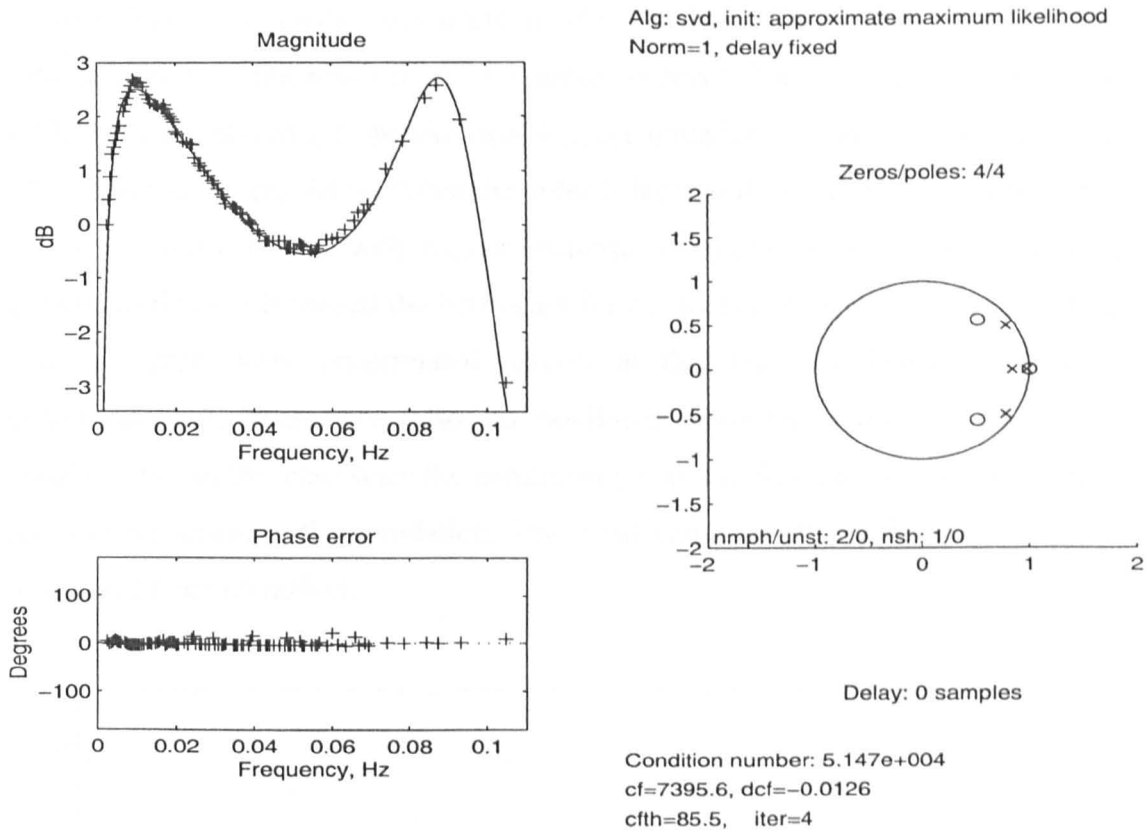

Figure 9.11. Example 2: Plot obtained using *elis* with model order 4/4 for L_2.

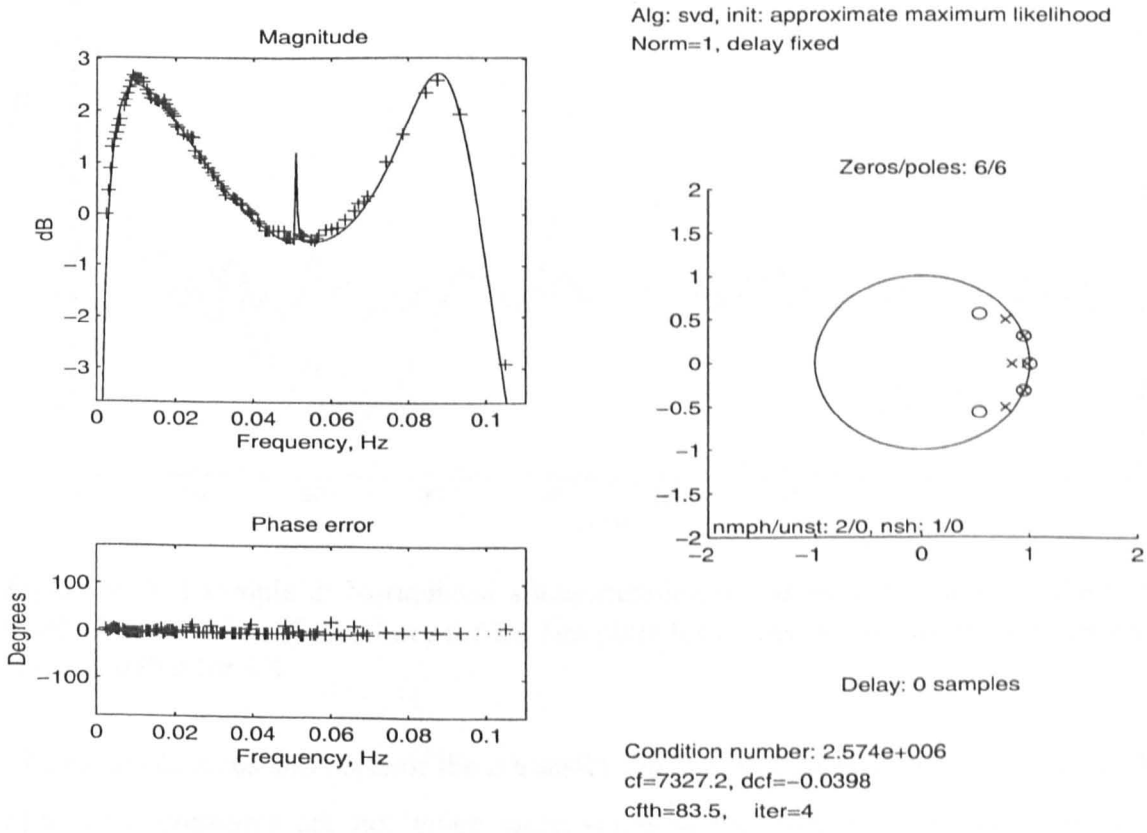

Figure 9.12. Example 2: Plot obtained using *elis* with model order 6/6 for L_2.

A final check was carried out using residual analysis. The plots of the normalised autocorrelation of the residuals R_{ee} for model orders 2/2 and 4/4 are shown in Figure 9.13. With an order of 2/2, the values of $|R_{ee}|$ are quite large, with some noticeable peaks other than that at zero delay. When the order is increased to 4/4, $|R_{ee}|$ decreases. There is no further drop in $|R_{ee}|$ with further increase in model order. Therefore this again confirmed that 4/4 is indeed the best order for L_2. Its narrow shape at lag zero suggests that the errors were uncorrelated (except at this lag) and hence, there was no undermodelling. There was also no nonlinear distortion caused by higher order nonlinearities in this case since the nonlinearity was strictly quadratic, and this had been taken into account in the simulation. The small values of $|R_{ee}|$ indicate that the correct model had been identified.

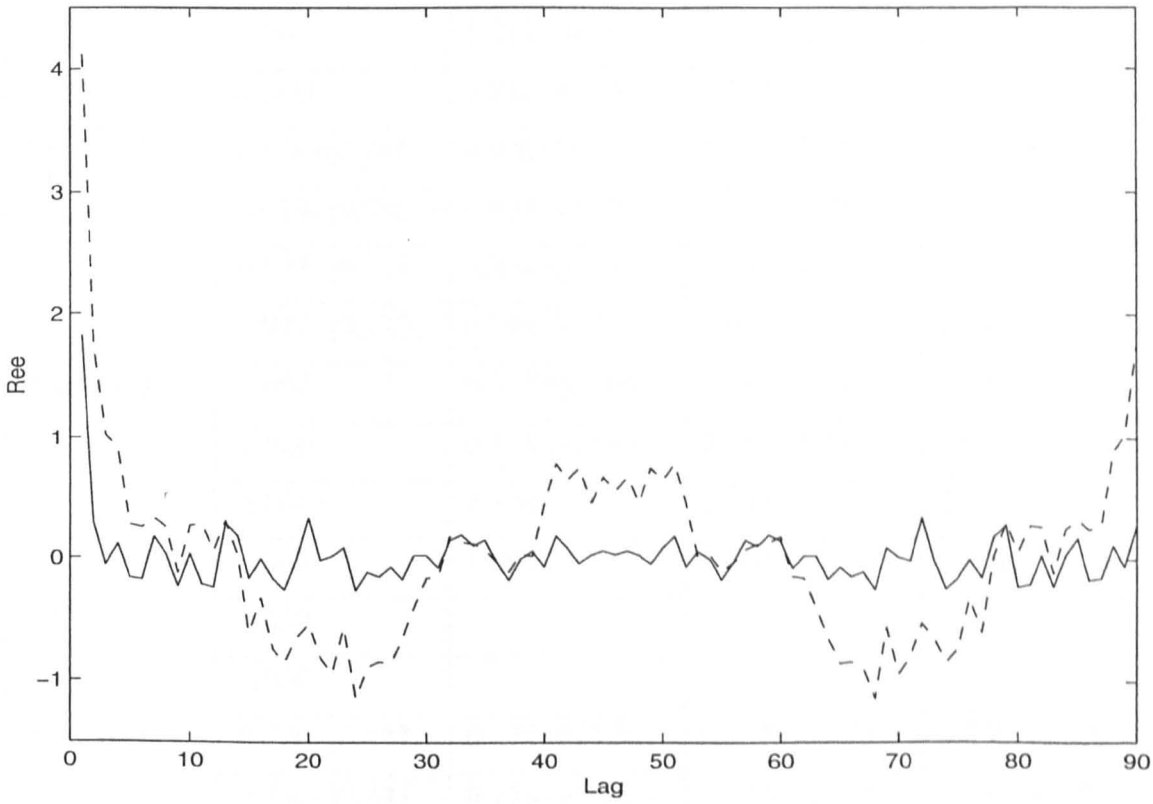

Figure 9.13. Example 2: Normalised autocorrelation of residuals for model orders 2/2 (dashed line) and 4/4 (solid line) for L_2. The plots for model orders above 4/4 were very similar to that for 4/4.

The values of zeros and poles of the z-transfer function estimates are given in Table 9.8. (The time constants are not given since some of the dynamics are underdamped.) Comparing the actual values with their estimates, it can be seen that the estimated poles

are reasonably close to the true ones. However, the estimated zeros are very far from the actual values. This is because the frequencies used in the estimation were much lower than the Nyquist frequency, resulting in the fact that no information of the frequency response was available at the frequencies close to the Nyquist frequency. The zeros were therefore placed by *elis* to achieve maximum effect in the frequencies of interest. It is also worth noting that while the actual order for L_2 is 6/6, there are two pairs of zeros and poles which are very close together which can be omitted. This is consistent with the analysis carried out when selecting the best order model.

Parameters	Actual	No noise	With noise	With distortion
Zeros of L_1	-1.000	0.687+j0.245	0.706+j0.223	0.668+j0.250
	-1.000	0.687- j0.245	0.706- j0.223	0.668- j0.250
	-1.000	1.212+j0.447	1.187+j0.402	1.219+j0.467
	-1.000	1.212- j0.447	1.187- j0.402	1.219- j0.467
Poles of L_1	0.939+j0.285	0.938+j0.283	0.937+j0.282	0.938+j0.284
	0.939- j0.285	0.938- j0.283	0.937- j0.282	0.938- j0.284
	0.947+j0.117	0.948+j0.117	0.948+j0.116	0.948+j0.117
	0.947- j0.117	0.948- j0.117	0.948- j0.116	0.948- j0.117
Zeros of L_2	-1.000	0.518+j0.566	0.514+j0.567	0.529+j0.554
	-1.000	0.518- j0.566	0.514- j0.567	0.529- j0.554
	-1.000	2.504	2.445	2.504
	1.000	1.004	1.004	1.004
	1.000			
	1.000			
Poles of L_2	0.770+j0.497	0.778+j0.502	0.777+j0.503	0.778+j0.500
	0.770- j0.497	0.778- j0.502	0.777- j0.503	0.778- j0.500
	0.845	0.840	0.842	0.839
	0.976	0.979	0.979	0.979
	0.999+j0.007			
	0.999- j0.007			

Table 9.8. Example 2: Estimated zeros and poles of L_1 and L_2.

It is also interesting to compare the results obtained for all the three cases examined (those without noise and distortion, with noise only, and with distortion only). There is a close match between these which again shows that the fitting of a parametric transfer function smoothes the effects of noise and input distortion as was found earlier in Section 9.6.1.

9.7 Conclusions

In this chapter, a new technique for identifying the linear subsystems of a Wiener-Hammerstein model was proposed. The underlying concept of this technique is the separation of the linear dynamics of the two linear subsystems through the symmetry properties of the second order Volterra kernel. The assumption of linearity between the measured points in the frequency domain is then used to calculate the linear dynamics of the linear subsystems by means of linear interpolation. This method was applied to two systems with different complexity and proven to give reasonably accurate estimates of the frequency responses in both cases. In one of the examples, the assumption of linearity was clearly violated in the gain response, and yet the technique remained robust. This was also true when noise and input distortion were added into the simulations.

The criteria for model selection were also investigated. The undesirable situations of undermodelling and overmodelling can be detected using statistical indicators, analysis of zeros and poles, analysis of frequency domain residuals by means of their normalised autocorrelation function, or a combination of these. Selecting a good model was found to be the main prerequisite to obtaining an accurate estimate of the transfer function from the nonparametric frequency response calculated using LIFRED. It was also shown that the fitting of a parametric transfer function smoothes the effects of noise and distortion.

Chapter 10

Extension of LIFRED to the Identification of Wiener-Hammerstein Models with Cubic Nonlinearity

10.1 Introduction

Identification of third order Volterra kernels of block-oriented models is a subject which has not been vigorously studied. There are papers on the subject, for example [140, 148, 162], but these are far fewer in number than those on the identification of second order kernels of these models. (The above topic is also considered in [59, 163, 164], but not specifically applied to block-oriented models.) The main reason is that the added dimension of the third order kernel makes analysis much more complicated and it is also difficult to present information visually. In this chapter, the identification using LIFRED (proposed in Chapter 9 for Wiener-Hammerstein models with quadratic nonlinearity) is extended to identify such models consisting of a cubic nonlinearity. The modifications to the algorithm are discussed and a simulation example is presented to illustrate the applicability of the method.

10.2 Modifications to the LIFRED Algorithm

10.2.1 Signal Design

The third order Volterra kernel of a Wiener-Hammerstein model, modified from the second order kernel in (9.4) and (9.5), is given by

$$|H_3(s_1,s_2,s_3)| = |L_1(s_1)|.|L_1(s_2)|.|L_1(s_3)|.|L_2(s_1+s_2+s_3)| \tag{10.1}$$

$$\angle H_3(s_1,s_2,s_3) = \angle L_1(s_1)+\angle L_1(s_2)+\angle L_1(s_3)+\angle L_2(s_1+s_2+s_3) \tag{10.2}$$

where L_1 and L_2 are the linear kernels of the first and second linear subsystems respectively. In order to extract this kernel from a perturbation test, good signal design is of great importance. To avoid Type II distortions [149, 150], a NID multisine [149] should be used. The signal would theoretically be more sparse than that used to identify the second order Volterra kernel since the number of possible contributions of the input frequencies is larger for a cubic nonlinearity compared with a quadratic nonlinearity. Also Type I contributions from points in the third order kernel where the frequencies

$f_1 = -f_2$ or $f_1 = -f_3$ or $f_2 = -f_3$ will fall at the input frequencies and hence cannot be separately measured [140].

In this chapter, a signal which consists of only odd harmonics is used, as was done in Chapter 9. This ensures that the measurement points of the third order kernel will not be corrupted by even order nonlinearities. The signal used, which was designed in [140] using an efficient search algorithm, is

$$fv = [241 \quad 451 \quad 663 \quad 877 \quad 1095 \quad 1319 \quad 1581 \quad 1817 \quad 2109 \quad 2347] \qquad (10.3)$$

It should be noted that the measurement of the third order Volterra kernel is much more sensitive to the effects of input signal distortion compared with the second order kernel. This is due to the fact that the number of possible combinations of the input frequencies increases approximately at the rate F^n, where F is the number of harmonics in the input signal and n is the order of the kernel. It should be stressed that this is not a weakness of the LIFRED method. Rather, it is a problem associated with the use of NID multisines for the measurement of the third order Volterra kernel, if input signal distortion could not be avoided. However, such problems are not encountered in a purely digital system, since the NID multisines can be exactly realised.

It is again assumed that the approximate bandwidths of the linear subsystems are known *a priori*, and that the power at the estimation lines are large enough to negate the effects of noise.

10.2.2 Calculation of the Frequency Response Gain of the Second Linear Subsystem

From equation (10.1), the gain at the output of the system is

$$|Z(s_1+s_2+s_3)| = |L_1(s_1)|.|L_1(s_2)|.|L_1(s_3)|.|L_2(s_1+s_2+s_3)|.|U(s_1)|.|U(s_2)|.|U(s_3)| \qquad (10.4A)$$

$$|Z(s_1+s_2-s_3)|=|L_1(s_1)|.|L_1(s_2)|.|L_1(-s_3)|.|L_2(s_1+s_2-s_3)|.|U(s_1)|.|U(s_2)|.|U(-s_3)| \tag{10.4B}$$

$$|Z(s_1-s_2+s_3)|=|L_1(s_1)|.|L_1(-s_2)|.|L_1(s_3)|.|L_2(s_1-s_2+s_3)|.|U(s_1)|.|U(-s_2)|.|U(s_3)| \tag{10.4C}$$

$$|Z(s_1-s_2-s_3)|=|L_1(s_1)|.|L_1(-s_2)|.|L_1(-s_3)|.|L_2(s_1-s_2-s_3)|.|U(s_1)|.|U(-s_2)|.|U(-s_3)| \tag{10.4D}$$

Using the symmetry property, $|L_1(s_2)| = |L_1(-s_2)|$, $|L_1(s_3)| = |L_1(-s_3)|$, $|U(s_2)| = |U(-s_2)|$ and $|U(s_3)| = |U(-s_3)|$. Hence, equation (9.8) now becomes

$$\frac{|Z(s_1+s_2+s_3)|}{|Z(s_1-s_2-s_3)|}=\frac{|L_2(s_1+s_2+s_3)|}{|L_2(s_1-s_2-s_3)|} \tag{10.5A}$$

$$\frac{|Z(s_1+s_2-s_3)|}{|Z(s_1-s_2-s_3)|}=\frac{|L_2(s_1+s_2-s_3)|}{|L_2(s_1-s_2-s_3)|} \tag{10.5B}$$

$$\frac{|Z(s_1-s_2+s_3)|}{|Z(s_1-s_2-s_3)|}=\frac{|L_2(s_1-s_2+s_3)|}{|L_2(s_1-s_2-s_3)|} \tag{10.5C}$$

Three different attenuation ratios α_1, α_2 and α_3 can then be defined for equations (10.5). In the actual program written, the smallest of the harmonics $|s_1 + s_2 - s_3|$, $|s_1 - s_2 + s_3|$ and $|s_1 - s_2 - s_3|$ is used in the denominators of equations (10.5). This harmonic will be stored in a vector called *minus*. (Note that $|s_1 + s_2 + s_3|$ is definitely larger than the above three harmonics.) The three remaining harmonics are stored in the vectors *plus1*, *plus2* and *plus3*. Therefore, the attenuation ratios are differently defined for different sets of harmonics s_1, s_2 and s_3, with α_1 linking *minus* and *plus1*, α_2 linking *minus* and *plus2*, and α_3 linking *minus* and *plus3*.

The values of α_1, α_2 and α_3 are calculated for all possible combinations of the input frequencies where $f_1 \neq f_2 \neq f_3$. For an input signal with F harmonics, this results in $(F^3 - 3F^2 + 2F)/6$ sets of α_1, α_2 and α_3. (There are F different choices for the first

harmonic, F-1 choices for the second harmonic and F-2 choices for the third harmonic. There are also six different ways to combine these three harmonics, resulting in a factor of six in the denominator.) All the entries in *minus* are then arranged in an increasing order. Likewise, the entries in *plus1*, *plus2* and *plus3* are reordered according to the order in *minus*.

In the interpolation algorithm, the first entry in *minus* according to the new order is set to unity. Next, the gains at *plus1*(1), *plus2*(1) and *plus3*(1) are calculated using *minus*(1), and $\alpha_1(1)$, $\alpha_2(1)$ and $\alpha_3(1)$. (The nomenclature used here is slightly different from that used in Chapter 9 because there are now three different attenuation ratios being considered (instead of only one in Chapter 9). In the present chapter, the number in the bracket denotes the kth element of a vector.) The gain at *minus*(2) is then estimated using linear interpolation between the two closest points to its left and right, which have already been estimated. *plus1*(2), *plus2*(2) and *plus3*(2) are calculated using *minus*(2), and $\alpha_1(2)$, $\alpha_2(2)$ and $\alpha_3(2)$. This continues until the gains at all the estimation lines are calculated. The program flow is similar to that in Chapter 9 and is therefore not summarised again here.

It should be noted that the algorithm attempts to estimate the lower frequencies first because the estimation lines are closer at these frequencies resulting in better accuracy using linear interpolation. Furthermore, most practical systems are lowpass which means that the measurement at lower frequencies is likely to be more accurate, and hence, the gain at these frequencies should be first calculated to ensure greater overall accuracy.

10.2.3 Calculation of the Frequency Response Gain of the First Linear Subsystem

When $s_1 = s_2 = s_3 = s$, (10.4A) can be written as

$$|Z(3s)| = |L_1(s)|^3 . |L_2(3s)| . |U(s)|^3 \tag{10.6}$$

The values of $|L_2(3s)|$ (except that with s equal to the highest and second highest harmonics) can be easily calculated by linear interpolation on the frequency response gain curve obtained using the procedure given in Section 10.2.2. The gain of L_1 is estimated using

$$|L_1(s)| = \frac{1}{|U(s)|} \cdot \sqrt[3]{\frac{|Z(3s)|}{|L_2(3s)|}} \tag{10.7}$$

However, the gains of L_1 at the highest and second highest harmonics have to be calculated using (10.4A), where s_1 and s_2 are set equal to the lowest and second lowest harmonics respectively, and s_3 is set equal to the harmonic at which the gain is to be calculated. Otherwise, these will have to be estimated through linear extrapolation, which may be much less accurate in the presence of noise.

10.2.4 Simultaneous Calculation of the Frequency Response Phases of the First and Second Linear Subsystems

For a third order kernel, equations (9.11) (for a second order kernel) can be extended to

$$\begin{aligned}\angle Z(|s_1+s_2+s_3|) &= \angle L_1(s_1)+\angle L_1(s_2)+\angle L_1(s_3)+\angle L_2(|s_1+s_2+s_3|)\\ &\quad +\angle U(s_1)+\angle U(s_2)+\angle U(s_3)\end{aligned} \tag{10.8A}$$

$$\begin{aligned}\angle Z(|s_1+s_2-s_3|) &= \angle L_1(s_1)+\angle L_1(s_2)-\angle L_1(s_3)+\angle L_2(|s_1+s_2-s_3|)\\ &\quad +\angle U(s_1)+\angle U(s_2)-\angle U(s_3)\end{aligned} \tag{10.8B}$$

$$\begin{aligned}\angle Z(|s_1-s_2+s_3|) &= \angle L_1(s_1)-\angle L_1(s_2)+\angle L_1(s_3)+\angle L_2(|s_1-s_2+s_3|)\\ &\quad +\angle U(s_1)-\angle U(s_2)+\angle U(s_3)\end{aligned} \tag{10.8C}$$

$$\begin{aligned}\angle Z(|s_1-s_2-s_3|) &= \angle L_1(s_1)-\angle L_1(s_2)-\angle L_1(s_3)+\angle L_2(|s_1-s_2-s_3|)\\ &\quad +\angle U(s_1)-\angle U(s_2)-\angle U(s_3)\end{aligned} \tag{10.8D}$$

Subtracting (10.8B) from (10.8A), and (10.8D) from (10.8C)

$$\begin{aligned}
&\angle L_2(|s_1+s_2+s_3|)-\angle L_2(|s_1+s_2-s_3|)\\
&\qquad=\angle Z(|s_1+s_2+s_3|)-\angle Z(|s_1+s_2-s_3|)-2\angle L_1(s_3)-\angle 2U(s_3)\\
&\qquad=\gamma_1 \qquad (10.9A)\\
&\angle L_2(|s_1-s_2+s_3|)-\angle L_2(|s_1-s_2-s_3|)\\
&\qquad=\angle Z(|s_1-s_2+s_3|)-\angle Z(|s_1-s_2-s_3|)-2\angle L_1(s_3)-\angle 2U(s_3)\\
&\qquad=\gamma_2 \qquad (10.9B)
\end{aligned}$$

where γ_1 and γ_2 are defined as the difference terms. It is possible to form three difference equations from (10.8) with another one relating γ_1 to γ_2. However this is not done since this will require either $\angle L_1(s_1)$ or $\angle L_1(s_2)$ to be known. Another point which needs careful consideration is that the harmonics $(s_1+s_2-s_3)$, $(s_1-s_2+s_3)$ and $(s_1-s_2-s_3)$ may be positive or negative and the corresponding signs of $\angle Z$ and $\angle L_2$ will have to be set accordingly. (Note that $(s_1+s_2+s_3)$ is always positive since only the positive harmonic numbers are considered. However, a modulus sign is still placed around it for the sake of consistency.) This means that γ_1 and γ_2 are differently defined for each set of s_1, s_2 and s_3. In some cases, these relate to the addition of two phases instead of the subtraction of one phase from the other. (Despite the above, these are still termed difference terms in order to stress the relation between these and γ in Chapter 9.)

The program used is very similar to that in Section 9.4.4 and therefore only the main modifications will be described here. The term $\angle L_1(s_3)$ with s_3 set to the lowest harmonic in the input is assigned an arbitrary value ϕ since the overall delay in the Wiener-Hammerstein model can be equally well attributed to the linear subsystems L_1 and L_2. This is in contrast to first assigning the phase at the third lowest harmonic in Section 9.4.4. This is because the third harmonic in (10.3) is quite large and it is preferable not to interpolate between this and the phase at dc. This means that the phase of the first linear subsystem at dc does not need to be known *a priori* (unlike in

Chapter 9). The values of $\angle L_2(s_1 + s_2 + s_3)$, $\angle L_2(s_1 + s_2 - s_3)$ $\angle L_2(s_1 - s_2 + s_3)$ and $\angle L_2(s_1 - s_2 - s_3)$ (with s_3 still set to the lowest harmonic) are estimated using a combination of linear interpolation and solving (10.9).

When this is done, the values of $\angle L_1(s_3)$ with s_3 set to the second, third and fourth lowest harmonics are solved simultaneously using (10.8A). All the rest of the phases of L_1 can then be estimated, again using (10.8A), and this completes the estimation of all the phases of L_1.

It is interesting to note that the phases at the remaining estimation lines of L_2 can be calculated directly using (10.8). There is also an option of doing this by linear interpolation. In the simulation in Section 10.3, the former was used in order to illustrate its effectiveness. (The latter was used in Section 9.6 and was already proven to work.) This will thus show that both methods work. It should be stressed that the technique using LIFRED is very flexible and many options are available regarding the choice of input harmonics, the sequence of estimation and the different estimation methods. This is an added advantage of using LIFRED.

10.3 Simulation Example

A simulation example was carried out where the first linear subsystem L_1 is a second order Chebyshev filter with 3dB of ripple in the passband and a cut-off frequency of 0.1 Hz. L_2 is a third order Chebyshev filter with 2dB of ripple in the passband and a cut-off frequency of 0.3 Hz. The input signal is a multisine of length $N = 16384$ with the harmonics given in (10.3). The sampling frequency was set at 1 Hz. The highest estimation lines used in estimating L_1 and L_2 were placed at 0.143 Hz and 0.430 Hz respectively. This gave sufficient excitation in the passbands of both L_1 and L_2.

The experiment was repeated with band-limited white noise added to the system output. The signal-to-noise ratio was approximately 30dB at the frequency band of interest.

It is worth noting that only an approximate knowledge of the bandwidths of the linear subsystems is required *a priori*. It may also pose a problem if the bandwidth of L_2 is very different from three times that of L_1 because then it is very difficult to set a sampling frequency which gives sufficient excitation in the passbands of both L_1 and L_2. In particular, if the bandwidth of L_1 is very much smaller than that of L_2, the signal power will be very much attenuated after passing through L_1, hence resulting in a low accuracy in the estimates of L_2. When the frequency response of L_1 is subsequently estimated from that of L_2, it will also have poor accuracy. The way to overcome this is to increase the signal power. However, it may or may not be practical depending on the specific application.

The third order Volterra kernels at Y and Z (in Figure 9.1(c)) are illustrated as volumetric slice plots (with the slice positions at $f_1 = f_2 = f_3 = 0$) in Figures 10.1 and 10.2 respectively. The symmetry property may be seen from these graphs. The frequency response of the linear subsystems L_1 and L_2 are shown in Figures 10.3 and 10.4 respectively.

It can be seen from Figures 10.3 and 10.4 that the estimates obtained without noise are extremely accurate, while those obtained in the presence of noise are still very good. This confirms the usefulness of the LIFRED technique. It is also clear from the above figures that the frequency response estimate of L_1 is more accurate than that of L_2. This is consistent with the statistical values of the errors E and E_S as defined in Section 9.6.1, and which are tabulated in Table 10.1.

Error estimates	No noise	With noise
E for L_1	$9.05 * 10^{-4}$	$3.01 * 10^{-2}$
E_S for L_1	$1.05 * 10^{-6}$	$1.16 * 10^{-3}$
E for L_2	$2.22 * 10^{-3}$	$5.23 * 10^{-2}$
E_S for L_2	$5.61 * 10^{-6}$	$3.98 * 10^{-3}$

Table 10.1. Values of E and E_S for L_1 and L_2.

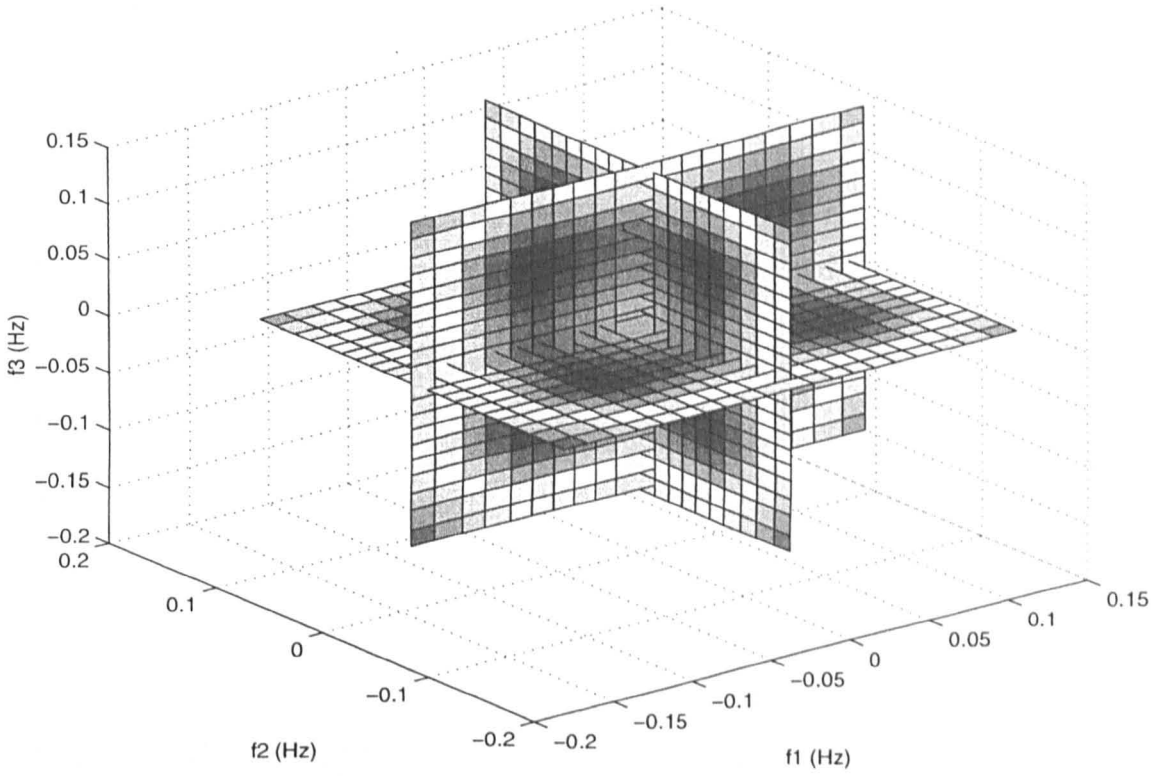

Figure 10.1. Third order Volterra kernel at Y.

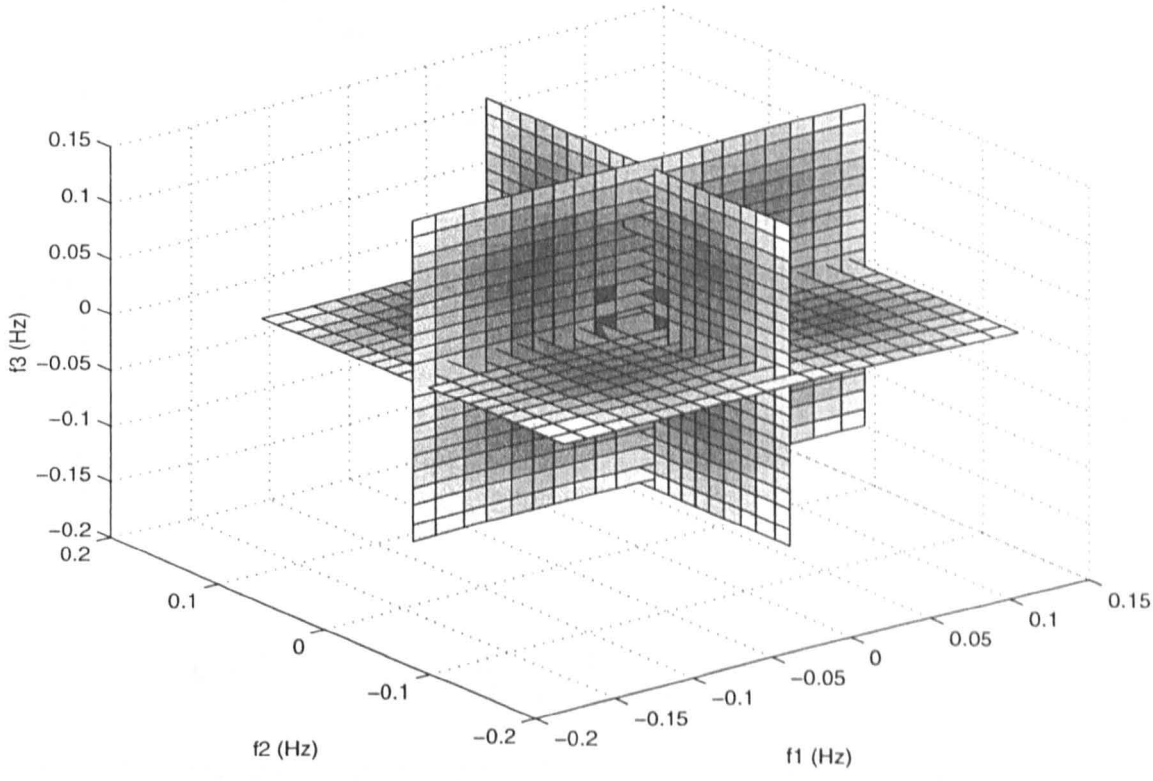

Figure 10.2. Third order Volterra kernel at Z.

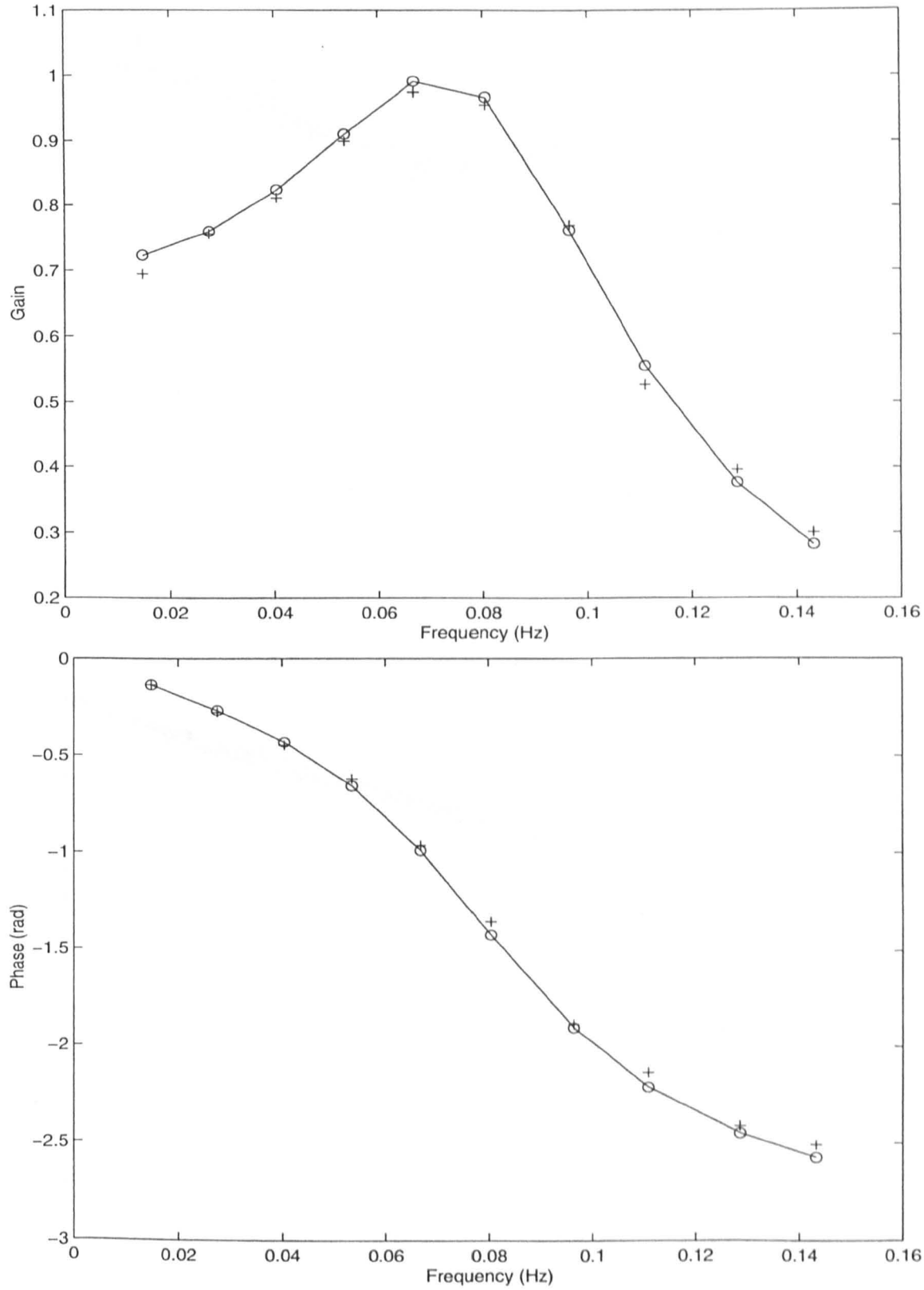

Figure 10.3. Frequency response gain (top) and phase (bottom) of L_1. Solid line: Actual values; Circles: Estimates obtained without noise; Plusses: Estimates obtained with noise.

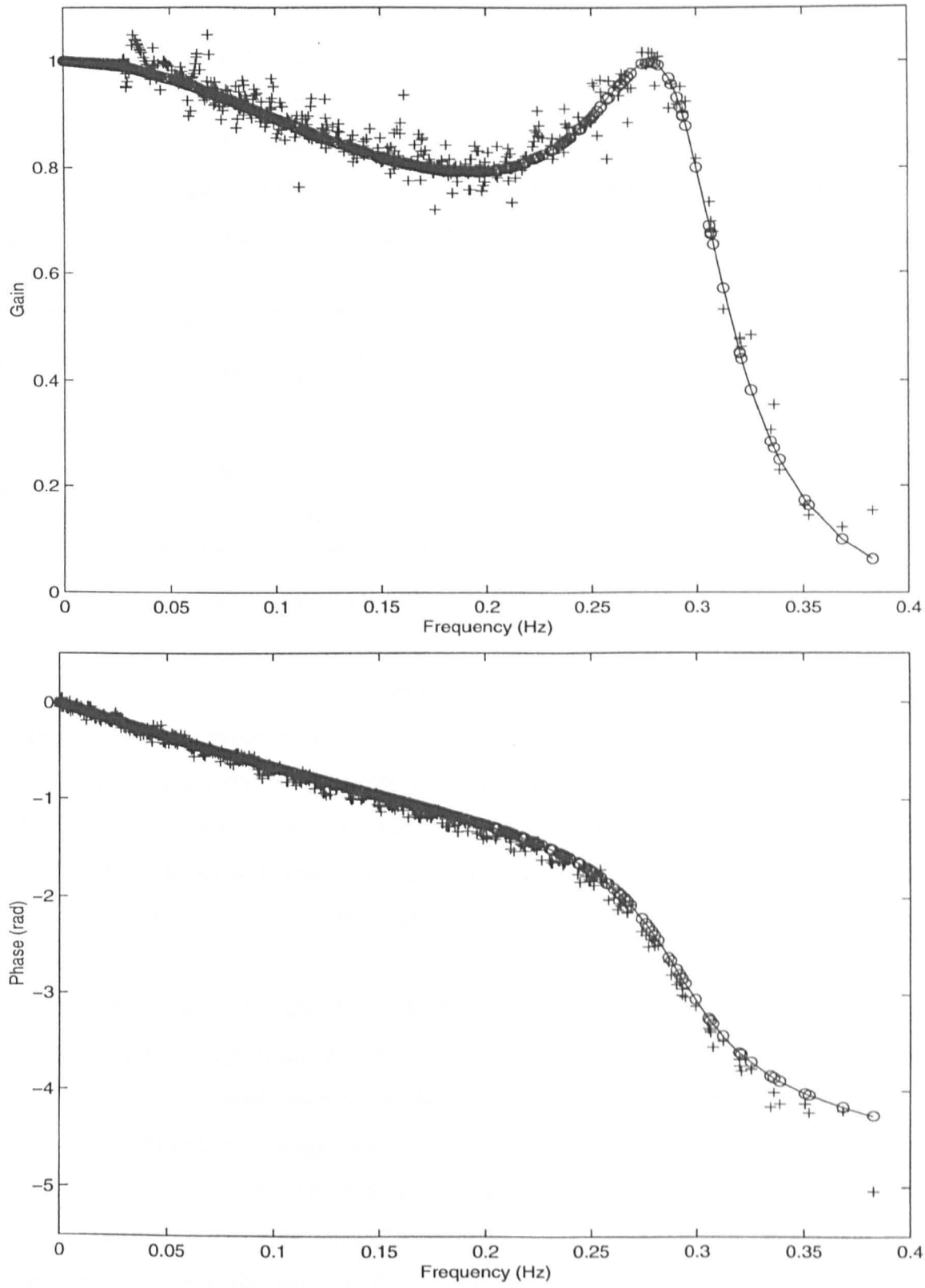

Figure 10.4. Frequency response gain (top) and phase (bottom) of L_2. Solid line: Actual values; Circles: Estimates obtained without noise; Plusses: Estimates obtained with noise.

The next step in the identification process was to obtain a parametric transfer function from the nonparametric frequency response. The statistical indicators discussed in Section 9.5.2.1 were used to select the best model order, in terms of the model accuracy and complexity. In the following analysis, the noise-free case was considered. Similar results could be obtained for the noisy case. The values for the indicators are given in Table 10.2 for the model orders considered for the linear subsystem L_1.

Model order	0/1	1/1	1/2	2/2	2/3	3/3
K	$1.39*10^7$	$6.01*10^6$	$2.03*10^4$	17.0	16.5	16.9
K_{min}	9.00	8.50	8.00	7.50	7.00	6.50
h_{mean}	0.455	0.299	0.017	$3.76*10^{-4}$	$3.77*10^{-4}$	$3.93*10^{-4}$
H_{mean}	0.555	0.682	0.717	0.715	0.715	0.715

Table 10.2. Statistical indicators for different model orders for L_1.

From Table 10.2, it can be seen that the values of the cost function K and the approximate mean model error h_{mean} drop drastically when the model order is increased to 2/2. These values then stay almost constant with further increase in the model order. Also, when the model order was set to 3/3, a zero-pole pair was formed at the position 1.054 in the z-plane, which was subsequently cancelled. Thus, it was concluded that the best model order for L_1 is 2/2. The graphical output from ELiS is shown in Figure 10.5.

For the subsystem L_2, the statistical indicators are given in Table 10.3. The cost function is seen to decrease significantly when the model order is increased to 3/3. The imaginary values for h_{mean} obtained when using the model orders 3/3 and 4/4 was caused by the variance values being too large (see Section 9.5.2.1). However, no variance values were available from the experiment as no averaging was carried out. When a 4/4 model was fitted, a zero-pole pair was formed at position 0.991 in the z-plane, which was then cancelled. Therefore, the best model order was chosen as 3/3. The graphical output from ELiS is shown in Figure 10.6.

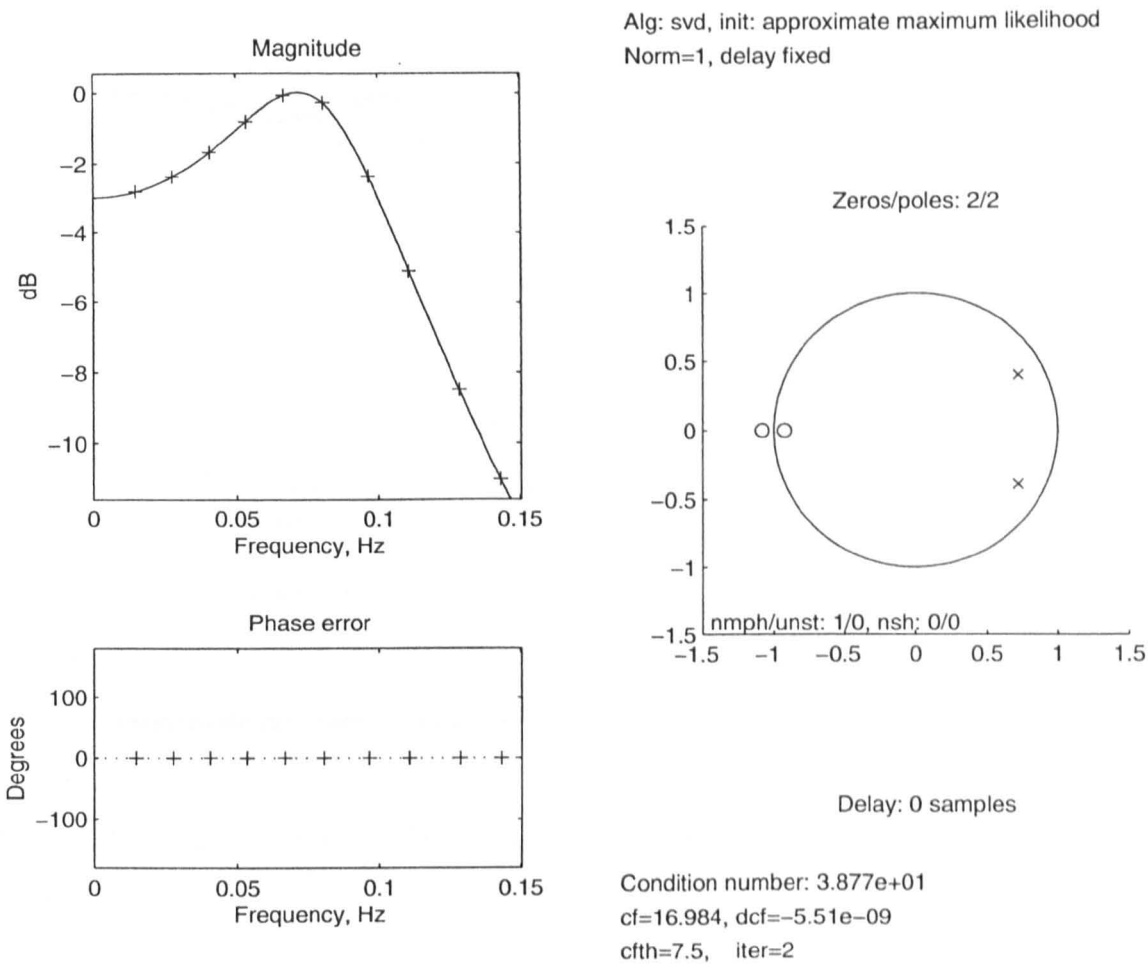

Figure 10.5. Plot obtained using *elis* with model order 2/2 for L_1.

Model order	1/1	2/2	3/3	4/4
K	$4.86*10^5$	$3.24*10^5$	56.7	52.3
K_{min}	479	478	477	476
h_{mean}	0.203	0.166	0.0060j	0.0060j
H_{mean}	0.877	0.881	0.883	0.883

Table 10.3. Statistical indicators for different model orders for L_2.

Since the best model orders for both L_1 and L_2 were clear from the above analysis, no further investigation was needed in this aspect. Also, it should be noted that these are in fact the correct model orders for the two linear subsystems. The parametric transfer functions were estimated using *elis* and the results obtained are shown in Table 10.4.

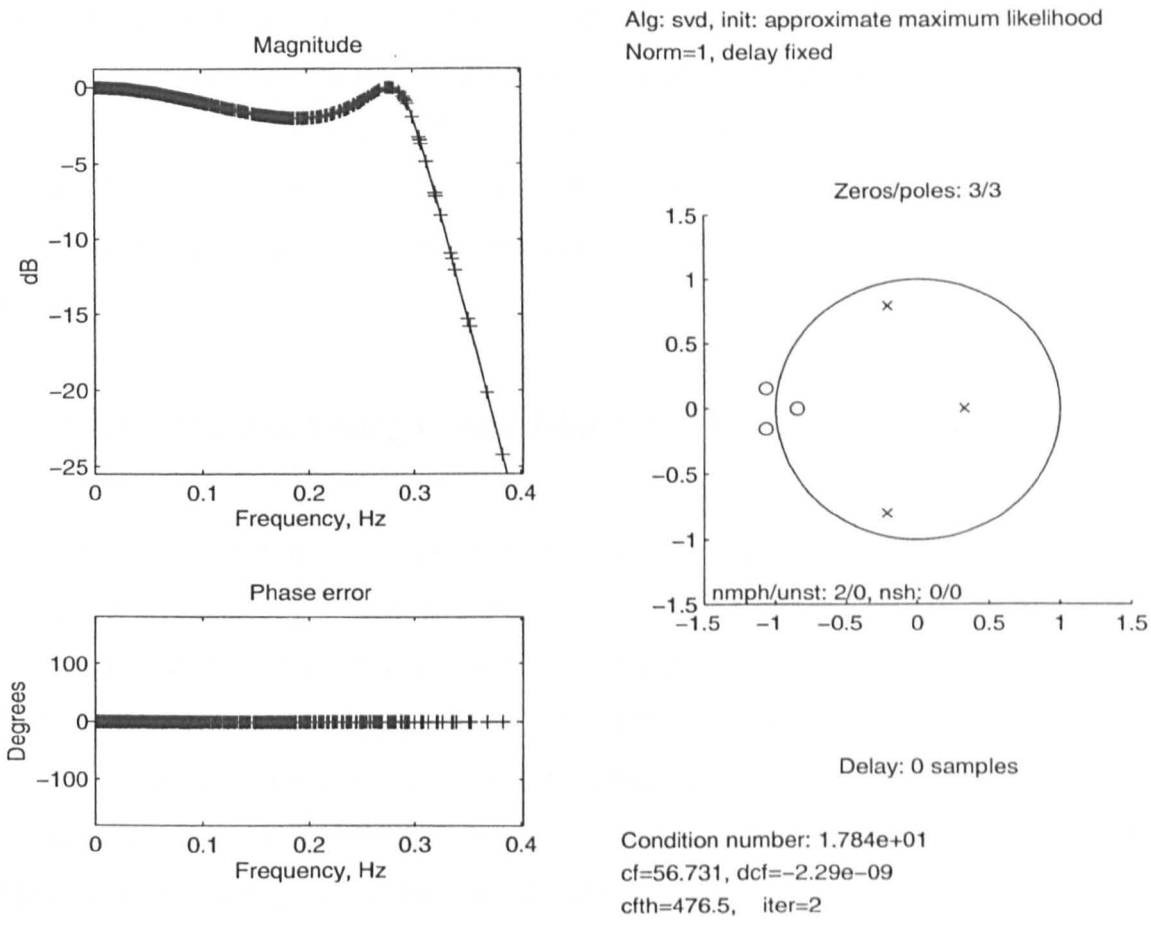

Figure 10.6. Plot obtained using *elis* with model order 3/3 for L_2.

Parameters	Actual	No noise	With noise
Zeros of L_1	-1.000	-1.084	-1.836
	-1.000	-0.927	-0.438
Poles of L_1	0.720+j0.393	0.721+j0.393	0.717+j0.394
	0.720- j0.393	0.721- j0.393	0.717- j0.394
Zeros of L_2	-1.000	-1.069+j0.155	-1.266+j0.540
	-1.000	-1.069- j0.155	-1.266- j0.540
	-1.000	-0.851	-0.595
Poles of L_2	-0.213+j0.798	-0.213+j0.798	-0.213+j0.792
	-0.213- j0.798	-0.213- j0.798	-0.213- j0.792
	0.327	0.326	0.329

Table 10.4. Estimated zeros and poles of L_1 and L_2.

From Table 10.4, the estimates obtained without noise are very accurate. For the case with noise added, the estimates of the poles are still reasonably accurate. However, the above is untrue for the values of the zeros. This is because the input signal excites a certain frequency band instead of the whole frequency spectrum. The zeros were therefore placed to give a better characterisation of the system at the frequencies of interest.

10.4 Estimation using Lines Distorted by Type I Distortion

In [140], it is shown that Type I contributions cause distortions in the third order kernel where the frequencies $f_1 = -f_2$ or $f_1 = -f_3$ or $f_2 = -f_3$ fall at the input frequencies and cannot be separately measured. These points are normally discarded and not used in the extraction of the frequency response of the linear subsystems. However, it will be now shown that these points can contribute to the estimation of the frequency response of L_1. To avoid confusion with the previous sections in the chapter, the harmonics used will be represented using fv_k, with k denoting the kth element of fv given in (10.3). Using the above notation,

$$
\begin{aligned}
|Z(fv_1)| &= |L_1(fv_1)|.|L_2(fv_1)|.\left|L_1(fv_1)^2 + L_1(fv_2)^2 + \ldots + L_1(fv_{10})^2\right|.|U(fv)|^3 \\
|Z(fv_2)| &= |L_1(fv_2)|.|L_2(fv_2)|.\left|L_1(fv_1)^2 + L_1(fv_2)^2 + \ldots + L_1(fv_{10})^2\right|.|U(fv)|^3 \\
&\ldots \\
|Z(fv_{10})| &= |L_1(fv_{10})|.|L_2(fv_{10})|.\left|L_1(fv_1)^2 + L_1(fv_2)^2 + \ldots + L_1(fv_{10})^2\right|.|U(fv)|^3
\end{aligned}
\tag{10.10}
$$

Hence

$$
\begin{aligned}
|L_{init1}(fv_1)| &= |L_1(fv_1)|\left|L_1(fv_1)^2 + L_1(fv_2)^2 + \ldots + L_1(fv_{10})^2\right| = |Z(fv_1)| / (|L_2(fv_1)|.|U(fv)|^3) \\
|L_{init2}(fv_2)| &= |L_1(fv_2)|\left|L_1(fv_1)^2 + L_1(fv_2)^2 + \ldots + L_1(fv_{10})^2\right| = |Z(fv_2)| / (|L_2(fv_2)|.|U(fv)|^3) \\
&\ldots \\
|L_{init10}(fv_{10})| &= |L_1(fv_{10})|\left|L_1(fv_1)^2 + L_1(fv_2)^2 + \ldots + L_1(fv_{10})^2\right| = |Z(fv_{10})| / (|L_2(fv_{10})|.|U(fv)|^3)
\end{aligned}
\tag{10.11}
$$

From (10.11), the values of $|L_1|$ are directly proportional to that of $|L_{init1}|$ and therefore they are first set equal to $|L_{init1}|$. They are then scaled such that

$$\left|L_1(fv_1)\right|\left|L_1(fv_1)^2 + L_1(fv_2)^2 + \ldots + L_1(fv_{10})^2\right| = \left|L_{init1}(fv_1)\right| \tag{10.12}$$

The values of $|L_1|$ obtained using the above method for the system considered in Section 10.3 are plotted in Figure 10.7. Unfortunately, a similar method could not be applied to calculate the phase because the input multisine signal does not have a constant phase. The values of the error are tabulated in Table 10.5.

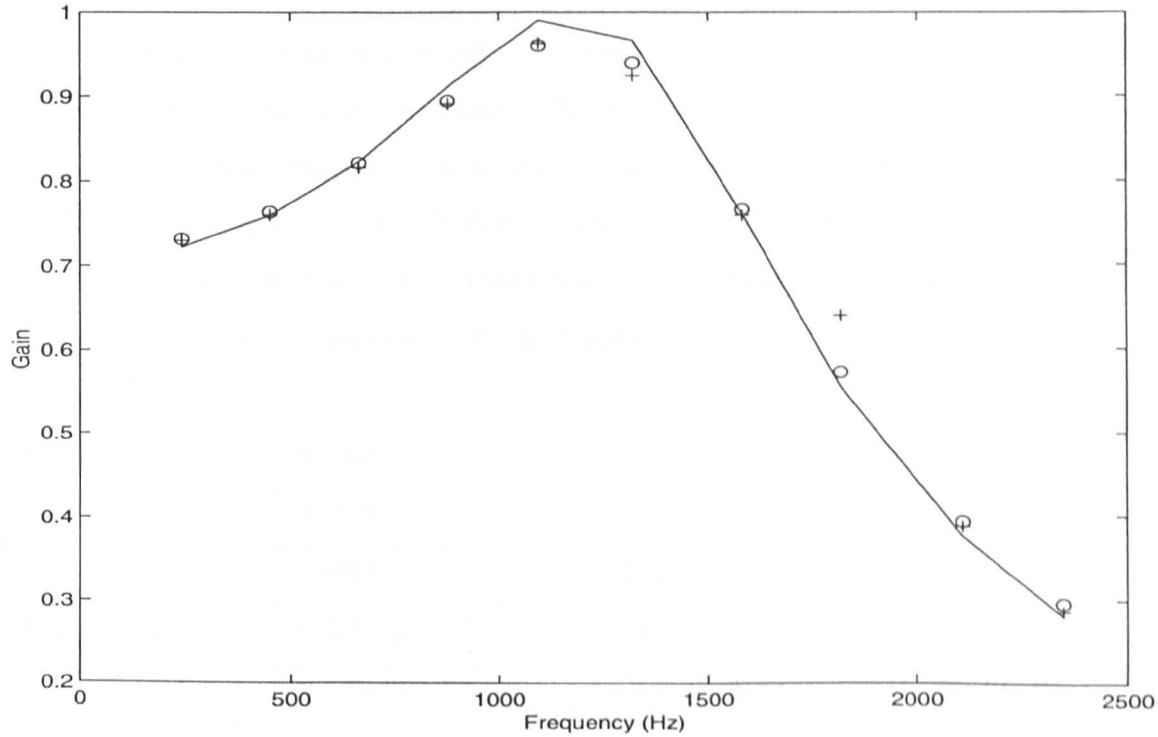

Figure 10.7. Frequency response gain of L_1 obtained from the distorted estimation lines. Solid line: Actual values; Circles: Estimates obtained without noise; Plusses: Estimates obtained with noise.

Error estimates	No noise	With noise
E for L_1	$1.45 * 10^{-2}$	$2.09 * 10^{-2}$
E_S for L_1	$2.90 * 10^{-4}$	$1.03 * 10^{-3}$

Table 10.5. Values of E and E_S for L_1 with the gain calculated from the estimation lines distorted by Type I distortion.

Comparing Table 10.1 with Table 10.5, it can be seen that the error is larger using the method just introduced. Despite this, the error is smaller than that in Table 10.1 in the presence of noise. (This is not obvious from Figure 10.7 if compared with Figure 10.3. However, the plots should be read in conjunction with that of the phase.) A possible explanation could be that the estimation lines have a much larger amplitude than those used in Section 10.3, due to several different combinations of the input harmonics. The discrete Fourier transform magnitude of the output (Z in Figure 9.1(c)) is illustrated in Figure 10.8. The larger amplitude of the estimation lines described above can be clearly seen in this figure.

As a final step, the parametric transfer function of L_1 was calculated and results obtained are tabulated in Table 10.6. From this table, it can be seen that the estimates are in fact less accurate than those in Table 10.4. This shows that there is a complicated relationship between the error obtained in the frequency response and that obtained in the transfer function. There is therefore no clear indication of which test frequencies should be used in the estimation of the gain response of L_1.

Parameters	Actual	No noise	With noise
Zeros of L_1	-1.000	-1.921	-2.473
	-1.000	-0.622	-0.563
Poles of L_1	0.720+j0.393	0.713+j0.393	0.709+j0.395
	0.720- j0.393	0.713- j0.393	0.709- j0.395

Table 10.6. Estimated zeros and poles of L_1 with the gain of L_1 estimated using the estimation lines corrupted with Type I distortion.

It should be also noted that the nonlinear block will introduce a scaling gain factor into the system. In the simulations conducted in this chapter (and in Chapter 9), the scaling factor has been set to unity. This is also assumed in all the equations given, in order to reduce the complexity involved. However, a non-unity scaling factor can be easily removed provided its value is known. This does not pose any problem since the information on the nonlinear block is assumed to be known *a priori*.

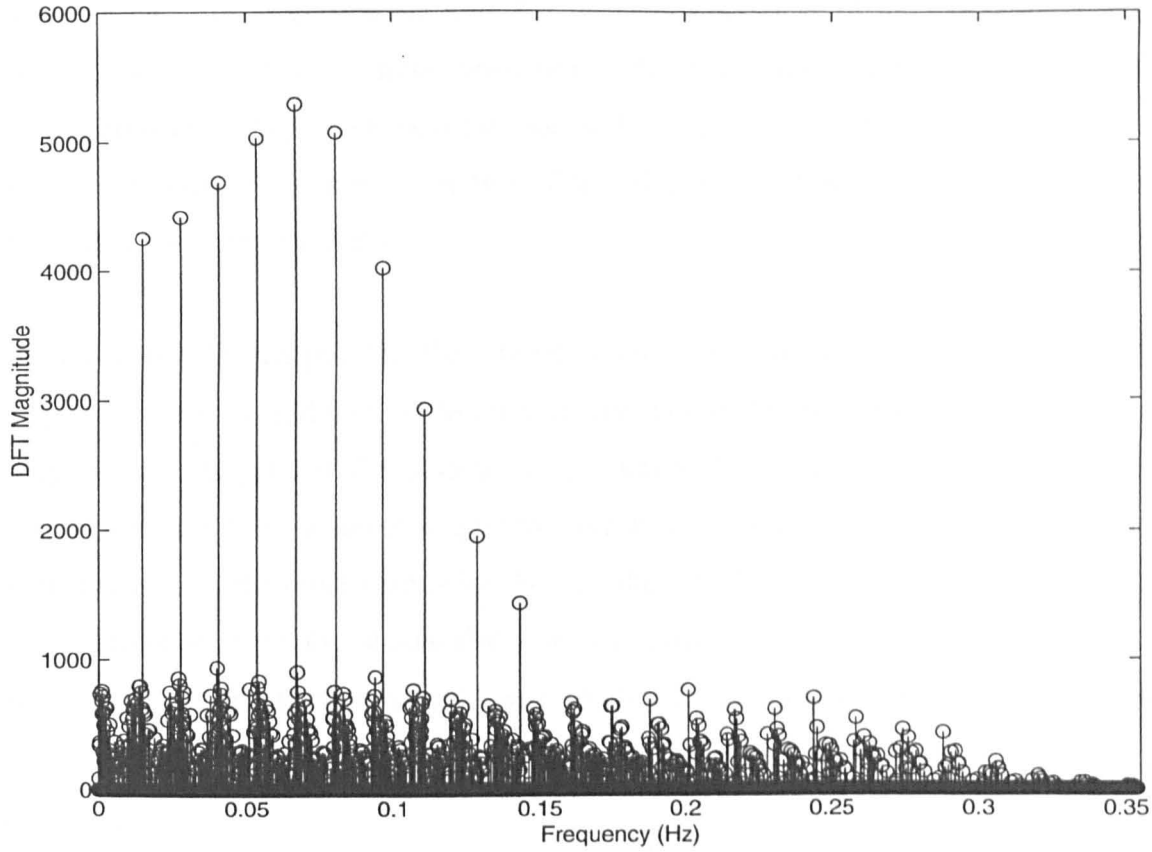

Figure 10.8. Discrete Fourier transform magnitude at the output of the Wiener-Hammerstein process.

10.5 Conclusions

In this chapter, the technique of LIFRED was extended from the second order Volterra kernel (in Chapter 9) to the third order kernel. The analysis is more complicated in the third order case but the basic ideas are very similar. A simulation example was conducted to illustrate the applicability of the method. High accuracy was achieved even in the presence of noise. However, unlike in the second order case, input distortion causes a major problem in the measurement of the third order kernel. This is because the number of possible combination of input harmonics which will result in a contribution at a particular harmonic in the output increases rapidly with the order of the kernel. This is in fact a measurement problem associated with the use of NID multisines and it affects any measurement and estimation technique that requires the use of such an input perturbation signal.

Model selection was carried out after the frequency response of the linear subsystems were obtained. For both these subsystems, the best model orders were correctly identified using values of the cost function and the approximate mean model error. The parametric transfer functions were then obtained using ELiS in the Frequency Domain System Identification Toolbox.

It is shown in this chapter that the estimation lines which are normally discarded due to Type I distortion could be used to estimate the gain of the first linear subsystem. This may be advantageous if the system is very noisy. The reason for this is that these estimation lines have a larger magnitude due to the contributions of several different combinations of the input harmonics. Hence, they are less susceptible to the effects of noise. However, it is also shown that a smaller error in the estimates of the frequency response does not necessarily result in a smaller error in the estimates of the transfer function.

Chapter 11

Conclusions

In the thesis, many aspects of system identification have been considered, with numerous application examples. Indeed, the main aim of the thesis was to investigate and hence promote the best practices in system identification. The importance of the above cannot be overstated as the application of identification increases rapidly, not only in engineering but also in other scientific areas such as physics, chemistry and biology. The reason for this is that identification allows complex real processes to be simplified considerably, while retaining their important static and dynamic characteristics.

Having said that, identification is not an 'easy' task and only a well designed identification experiment will yield accurate knowledge of the system under consideration. In this thesis, the main areas in identification, which are the experimental design, model construction and parameter estimation were investigated. It was hoped that the results of this study would promote the best practices in the general framework of identification.

The main conclusions from the four sub-sections of the thesis will now be discussed in turn. These are periodic perturbation signal design, processes with direction-dependent dynamics, autotune control of processes with significant dead time and the identification of Wiener-Hammerstein models. Indications to areas of future research will be given where appropriate.

11.1 Periodic Perturbation Signal Design

The many advantages of periodic perturbation signals have been long known and well documented in the literature. In Chapter 2, the design of binary and multi-level pseudo-random signals were considered. For a linear system and a nonlinear system with a Wiener structure, a binary signal is persistently exciting, and should be used when possible because it has maximum signal power within amplitude constraints. It was shown that besides the well known MLB signals, there are other classes of binary or near-binary pseudo-random signals which have the same autocorrelation properties (or

nearly the same, for the case of near-binary signals) as the MLB signals. A MATLAB program was written to generate these classes of signal since it is useful to have at hand many different signals to choose from. This alleviates the problem of being restricted in terms of signal length due to the MLB signals having a possible length that increases approximately at a power by two. In particular, the QRB and QRT signals were found to have many more possible periods in which these signals are available.

For the identification of a nonlinear model with a Hammerstein structure, a binary signal is not persistently exciting. A multi-level signal must be used and the design of signals generated from a Galois field was investigated. It was found that the input-output relationship of the system can be described by a Vandermonde matrix, which greatly facilitates analysis. The importance of signal design can be clearly seen in this chapter and some measures of signal quality were also investigated, both for applications in linear and nonlinear systems. In general, it is desirable to maximise the signal power at the specified harmonics within amplitude constraints and minimise the sensitivity of the system with respect to noise, bearing in mind that persistent excitation is a requirement, not an added advantage.

11.2 Processes with Direction-dependent Dynamics

The pseudo-random signals designed in Chapter 2 and multisine signals were applied to the detection of direction-dependent dynamics in processes. It was found that only signals based on maximum length sequences give coherent patterns in the input-output crosscorrelation function due to shift-and-add and shift-and-subtract properties. The positions of the coherent peaks depend on the actual signal used, and not only on the signal length. Some signals have these peaks further away from the main linear peak thus making their detection much easier. This emphasised the need to have many choices of signals, even those within the same class, as was stated in the previous section. It was also interesting to find that such patterns are not present in either the gain or phase of the frequency response. It was concluded that the MLB signal is the best class of signal to use for the detection of the departure from linearity, since in this case,

a binary signal is persistently exciting and it maximises the signal power within the specified peak-to-peak amplitude.

While an MLB signal should be used for the above purpose, the corresponding inverse-repeat signal should be applied to identify the combined linear dynamics of the system. This is because the use of an inverse-repeat signal allows odd and even order terms to be separated at the system output, thus making it possible to eliminate the effects of even order nonlinearities resulting in a higher accuracy and consistency of the estimates obtained. This clearly brought out the fact that the optimal perturbation signal is application dependent, and it depends on the purpose of the identification as much as on the process which it is applied to.

The estimation of the combined linear dynamics presented in Chapter 4 was conducted both in time and frequency domains, with the help of functions available in the System Identification Toolbox and the Frequency Domain System Identification Toolbox in MATLAB. It was found that for a first order process perturbed with a binary signal, the combined time constant is closer to that of the smaller time constant. However, theoretical results could not be obtained for second order processes, due to the direction of the output being not necessarily in the same direction as the input signal (assuming that the input signal levels are symmetrical about zero).

The possibility of deriving the theoretical expressions for the output and the input-output crosscorrelation function was further utilised in Chapter 5 where the modelling of such processes was undertaken. The nonlinear dynamics of the direction-dependent process was 'converted' into a static nonlinearity by modelling the process using a Wiener model. The parameters of the Wiener model were obtained using the Optimization Toolbox in MATLAB. This has wide applicability in the industry as a static nonlinearity is much easier to handle and more amenable to analysis. Further work is necessary in this area to find ways to extend the analysis to that for a first order process perturbed with a ternary signal, and a second (or higher) order process perturbed with a binary signal.

The modelling of direction-dependent dynamics using a neural network architecture was investigated in Chapter 6, in line with the growing popularity of neural networks. Research into the many different architectures has increased tremendously over the last decades amidst the believe that such networks can model many types of nonlinearities provided they are sufficiently trained and have sufficient number of neurons. Also, the approach has generated a lot of interest due to it being not so 'mathematical', with many of the parameters being 'hidden' within the network. Thus the aim of this chapter was to compare the performance of the neural network model with that of the Wiener model in trying to match the output of a direction-dependent process. It was found that both models have merits of their own, and the better model to use depends on the dynamics of the actual process and the perturbation signal applied.

An experiment on a real process was conducted using an electronic nose, where the direction-dependent characteristics were caused by the different adsorption and desorption rates of the chemical odour on the MOS sensor surface. The identification process was carried out from the detection of the departure from linearity to the identification of the combined linear dynamics to the modelling using the Wiener model and the neural network model. Thus, the ideas proposed in Chapters 3 to 6 were put to test on a real process and were found to work remarkably well. Interesting results were obtained throughout the experiment which proved the sound basis of the theoretical analysis given in these chapters.

In Chapter 7, the control of direction-dependent processes using the well known PID controller was investigated. In industry, the controller parameters are normally set according to the direction of the process output. Hence, two sets of controller parameters are needed for effective control of the process. However in this chapter, a single set of controller parameters tuned based on the combined linear dynamics of the process was used instead. The performance of the controller was studied and it showed the danger of not recognising the existence of such a departure from linearity. When the gains in the upward and downward directions are different, limit cycles may occur. A moving

averager was added in between the PID controller and the process to smooth the output of the controller in order to alleviate the problem.

11.3 Autotune Control of Processes with Significant Dead Time

In Chapter 8, the autotune control of processes with significant dead time was considered. It was shown that the identification using multisines allows the process to be identified in closed-loop. This has the advantage that it does not require interruption of normal closed-loop operation, in order to allow for a separate identification test. It is thus time-saving to identify the process in closed-loop. In addition, the parameters obtained in this manner will more closely resemble those in the intended operation of the process which is very desirable. A multisine signal which enables the user to completely specify the frequency spectrum is in this case a great blessing and this again shows the importance of signal design in every identification experiment.

In this chapter, the dead time was compensated using a Smith predictor. The parameters of the PI controller was set according to the model of the dead time free part of the process. A derivative term was not included as this slows down the transient response for a delay free process. (The dead time was already 'taken out' of the control loop by the Smith predictor.) The controller performance was compared between the above and a PI controller without Smith predictor (the derivative term is ineffective in the control of processes with long dead time), and that of the Dahlin controller. It was found that in general, the Smith predictor with PI control has the best overall performance. An experiment on a hot-air flow device, which was conducted using the Real-Time Windows Target in SIMULINK, verified the results obtained through simulation. In fact, the results were extremely interesting considering that the ratio of the dead time to the time constants of the process was less than unity. This shows that dead time compensation is useful even if the dead time is not all that significant (compared to the process dynamics). Also, due to the many different possible performance criteria for a

controller such as the step response, load disturbance rejection, settling time, robustness and control signal amplitude, there is scope for further research in this area.

11.4 Identification of Wiener-Hammerstein Models

Block-oriented models such as the Wiener, Hammerstein, Wiener-Hammerstein and the much less known Hammerstein-Wiener models are increasingly popular as alternatives to directly dealing with the Volterra series expansions for nonlinear systems. Even though these models are less complex, they are by no means simple. In Chapter 9, Wiener-Hammerstein models were considered, where effective identification methods were lacking, and most of them require long testing times and a huge computational burden. Thus in this chapter, the aim was to propose a novel technique which does not inherit the disadvantages mentioned above. This technique is based on the extraction of the symmetry properties of the Volterra kernel for a Wiener-Hammerstein structure with quadratic nonlinearity. The estimation of linear dynamics of the linear subsystems are carried out using linear interpolation. The method was shown to work well regardless of whether the frequency response gain and phase curves resemble a straight line, which is the implicit requirement for the use of linear interpolation. Also, the technique is robust to the presence of noise and input signal distortion.

The success of the technique proposed was partly due to the availability of a class of multisine signals with no interharmonic distortion. This class of signal possesses the useful property which allows the terms in the Volterra kernel to be separated. Hence, the importance of signal design was again highlighted here and this is true in any other application.

In most literature, the identification method is proposed for a nonlinear system with quadratic nonlinearity and is implicitly assumed to work for those with higher order nonlinearities. In Chapter 10, such an extension to a cubic nonlinearity was carried out and the modifications to the equations and programs used were clearly explained. A simulation example was also conducted. It was found that the technique is applicable

even under noisy conditions. Unfortunately, input signal distortion will cause problems due to the many possible combinations of the input harmonics. It was also shown that the method has a high flexibility, and the user has, at times, a wide choice regarding the sequence of harmonics to be estimated and the equations to use. The best choices for the above could not be stated definitely, as these would depend on the particular application. After all, many of the decisions made in an identification experiment are application dependent. However, it could be said that this is an interesting area which warrants further research. The use of other interpolation techniques, such as those based on splines could yield promising results.

11.5 Concluding Remarks

In the thesis, the many facets of system identification have been discussed, encompassing areas of experimental design, model construction and parameter estimation. The four main topics covered were periodic perturbation signal design, processes with direction-dependent dynamics, autotune control of processes with significant dead time and the identification of Wiener-Hammerstein models. Throughout the thesis, the importance that the identification design be tailored to the specific application was stressed and clearly brought out in the examples illustrated. The main aim of the thesis, which was to investigate and hence promote the best practices in system identification, was successfully achieved. Directions requiring further research were also given.

References

[1] VAN DEN BOS, A. : 'Periodic test signals -properties and use', *in* GODFREY, K. (Ed.) : 'Perturbation Signals for System Identification', (Prentice Hall, UK, 1993), Chap. 4.

[2] DARNELL, M. : 'Periodic and non-periodic, binary and multi-level pseudo-random signals', *in* GODFREY, K. (Ed.) : 'Perturbation Signals for System Identification', (Prentice Hall, UK, 1993), Chap. 5.

[3] GODFREY, K. : 'Introduction to perturbation signals for frequency-domain system identification', *in* GODFREY, K. (Ed.) : 'Perturbation Signals for System Identification', (Prentice Hall, UK, 1993), Chap. 2.

[4] GODFREY, K. R., BARKER, H. A. and TUCKER, A. J. : 'Comparison of perturbation signals for linear system identification in the frequency domain', *IEE Proc. - Control Theory Appl.*, 1999, **146** (6), pp. 535 - 548.

[5] KOLLÁR, I. : 'Frequency Domain System Identification Toolbox for use with MATLAB', (The MathWorks Inc., Natick, Mass., 1994).

[6] McCORMACK, A. S., GODFREY, K. R. and FLOWER, J. O.: 'Design of multilevel multiharmonic signals for system identification', *IEE Proc. - Control Theory Appl.*, 1995, **142** (3), pp. 247 - 252.

[7] ZIERLER, N. : 'Linear recurring sequences', *J. Soc. Ind. Appl. Math.*, 1959, **7**, pp. 31 - 48.

[8] BARKER, H. A. and ZHUANG, M. : 'GALOIS a program for generating pseudo-random perturbation signals', *Proc. 11th IFAC Symposium on System Identification*, Kitakyushu, Japan, 1997, p. 1641.

[9] GODFREY, K. : 'Introduction to perturbation signals for time-domain system identification', *in* GODFREY, K. (Ed.) : 'Perturbation Signals for System Identification', (Prentice Hall, UK, 1993), Chap. 1.

[10] EVERETT, D. : 'Periodic digital sequences with pseudonoise properties', *G. E. C. Journal of Science and Technology*, 1966, **33**, pp. 115 - 126.

[11] BILLINGS, S. A. and FAKHOURI, S. Y. : 'Identification of nonlinear systems using the Wiener model', *Electronics Letters*, 1977, **13** (17), pp. 502 - 504.

[12] ZHU, Y. : 'Identification of Hammerstein models for control', *Proc. IEEE Conference on Decision and Control*, **1**, Tampa, Florida, USA, 1998, pp. 219 - 220.

[13] NORDSJO, A. E. : 'Recursive prediction error algorithms for joint time delay and parameter estimation of certain classes of nonlinear systems', *Proc. IEEE Conference on Decision and Control*, **3**, Tampa, Florida, USA, 1998, pp. 3429 - 3438.

[14] VANDERSTEEN G. and SCHOUKENS, J. : 'Measurement and identification of nonlinear systems consisting of linear dynamic blocks and one static nonlinearity', *IEEE Trans.*, 1999, **AC - 44** (6), pp. 1266 - 1271.

[15] KWAK, B. J., YAGLE, A. E. and LEVITT, J. A. : 'Nonlinear system identification of hydraulic actuator friction dynamics using a Hammerstein model', *Proc. IEEE Conference on Acoustics, Speech and Signal Processing*, **4**, Seattle, USA, 1998, pp. 1933 - 1936.

[16] EMARA-SHABAIK, H. E. and MOUSTAFA, K. A. F. : 'Non-parametric identification of generalized Hammerstein models', *International Journal of Systems Science*, 1998, **29** (1), pp. 91 - 93.

[17] SUTTER, E. E. : 'A practical nonstochastic approach to nonlinear time-domain analysis', *in* MARMARELIS, V. Z. (Ed.) : 'Advanced Methods of Physiological System Modelling', (University of Southern California, Los Angeles, 1987).

[18] BARKER, H. A., GODFREY, K. R. and TUCKER, A. J. : 'Nonlinear system identification with multi-level perturbation signals', *12th IFAC Symposium on System Identification SYSID*, Santa Barbara, California, 2000, Session FrPM3, Paper No. 3.

[19] MARMARELIS, P. Z. and MARMARELIS, V. Z. : 'Analysis of Physiological Systems - The White Noise Approach', (Plenum, New York, 1978).

[20] SHI, Y. : 'System Identification with Maximum Length Sequences and Applications in Brainstem Auditory Evoked Responses', PhD. Dissertation, University of Wisconsin-Madison, 1990.

[21] PETERSON, W. W. : 'Error-correcting Codes', (M. I. T. Press, 1961).

[22] MARSH, R. W. : 'Table of Irreducible Polynomials over GF(2) through Degree 19', (NSA, 1957).

[23] WATSON, E. J. : 'Primitive polynomials (modulo-2)', *Mathematics of Computing*, 1962, **16**, pp. 368 - 369.

[24] BRIGGS, P. A. N., HAMMOND, P. H., HUGHES, M. T. G. and PLUMB, G. O. : 'Correlation analysis of process dynamics using pseudo-random binary test perturbations', *Proc. Inst. Mech. Engrs*, 1965, **179**, Part 3H, pp. 37 - 51.

[25] GODFREY, K. R. and BRIGGS, P. A. N. : 'Identification of processes with direction-dependent dynamic responses', *IEE Proc. - Control & Science*, 1972, **119** (12), pp. 1733 - 1739.

[26] GODFREY, K. R. and MOORE, D. J. : 'Identification of processes having direction-dependent responses, with gas-turbine engine applications', *Automatica*, 1974, **10** (5), pp. 469 - 481.

[27] TSAO, S. H. : 'Generation of delayed replicas of maximal-length linear binary sequences', *IEE Proc.*, 1964, **111**, pp. 1803 - 1806.

[28] VOLTERRA, V. : 'Theory of Functionals and of Integral and Integrodifferential Equations', (Blackie, London, 1930).

[29] BARKER, H. A. and OBIDEGWU, S. N. : 'Effects of nonlinearities on the measurement of weighting functions by cross-correlation using pseudorandom signals', *IEE Proc.*, 1973, **120** (10), pp. 1293 - 1300.

[30] GODFREY, K. R. and MURGATROYD, W. : 'Input transducer errors in binary crosscorrrelation experiments', *IEE Proc.*, 1965, **112** (3), pp. 565 - 573.

[31] VINOGRADOV, I. M. : 'An Introduction to the Theory of Numbers', (Pergamon Press, 1955).

[32] NOWAK, R. D. and VAN VEEN, B. D. : 'Random and pseudorandom inputs for Volterra filter identification', *IEEE Trans.*, 1994, **SP - 42** (8), pp. 2124 - 2135.

[33] GAUTSCHI, W. and INGLESE, G. : 'Lower bounds for the condition number of Vandermonde matrices', *Numer. Math.*, 1988, **52**, pp. 241 - 250.

[34] BARKER, H. A. and ZHUANG, M. : 'Design of pseudo-random perturbation signals for frequency-domain identification of nonlinear systems', *Proc. 11th IFAC Symposium on System Identification*, Kitakyushu, Japan, 1997, pp. 1635 - 1640.

[35] GODFREY, K. R. : 'Three-level m sequences', *Electronics Letters*, 1966, **2** (7), pp. 241 - 243.

[36] BARKER, H. A. : 'Design of multi-level pseudo-random signals for system identification', *in* GODFREY, K. (Ed.) : 'Perturbation Signals for System Identification', (Prentice Hall, UK, 1993), Chap. 11.

[37] CHURCH, R, : 'Tables of irreducible polynomials for the first four prime moduli', *Ann. Math.*, 1935, **36**, p. 198.

[38] BALZA, C. FROMAGEOT, A. and MANIERE, M. : 'Four-level pseudo-random sequences', *Electronics Letters*, 1967, **3**, pp. 313 - 315.

[39] BRIGGS, P. A. N. and GODFREY, K. R. : 'Autocorrelation function of a 4-level *m*-sequence', *Electronics Letters*, 1968, **4** (11), pp. 232 - 233.

[40] TURNER, P., MONTAGUE, G. and MORRIS, J. : 'Nonlinear and direction-dependent dynamic process modelling using neural networks', *IEE Proc. - Control Theory Appl.*, 1996, **143** (1), pp. 44 - 48.

[41] ROSENQVIST, F., ERIKSSON, K. and KARLSTRÖM, A. : 'Modelling a thermomechanical wood-chip refiner', *Proc. IASTED Modelling, Identification and Control Conference*, Innsbruck, Austria, 2001, pp. 763 - 768.

[42] ROSENQVIST, F., KARLSTRÖM, A. and ERIKSSON, K. : 'Parameter estimation in processes with direction-dependent dynamics', *Proc. 3rd International Conference on Identification in Engineering Systems*, Swansea, UK, 2002, pp. 68 - 77, LEES, A. W., PRELLS, U. and NORTON, J. P. (Ed.), (Institute of Physics, 2002).

[43] SCHROEDER, M. R. : 'Synthesis of low-peak factor signals and binary sequences with low autocorrelation', *IEEE Trans.*, 1970, **IT - 16** (1), pp. 85 - 89.

[44] VAN DER OUDERAA, E., SCHOUKENS, J. and RENNEBOOG, J. : 'Peak factor minimization using a time-frequency swapping algorithm', *IEEE Trans.*, 1988, **IM - 37** (1), pp. 144 - 147.

[45] LJUNG, L. : 'System Identification Toolbox for use with MATLAB', (The MathWorks Inc., Natick, Mass., 2000).

[46] ISERMANN, R. and BAUR, U. : 'Two-step process identification with correlation analysis and least squares parameter estimation', *J. Dynam. Syst. Meas. Cont.*, 1974, **96** (4), pp. 426 - 432.

[47] ISERMANN, R., BAUR, U., BAMBERGER, W., KNEPPO, P. and SIEBERT, H. : 'Comparison of six on-line identification and parameter estimation methods', *Automatica*, 1974, **10** (1), pp. 81 - 103.

[48] LJUNG, L. : 'System Identification - Theory for the User', (Prentice Hall, USA, 1987).

[49] HANNAN, E. J. : 'The identification and parametrization of ARMAX and state space forms', *Econometrica*, 1976, **44**, pp. 713 - 723.

[50] CAINES, P. E. : 'On the asymptotic normality of instrumental variable and least squares estimators', *IEEE Trans.*, 1976, **AC - 21**, pp. 598 - 600.

[51] CAINES, P. E. : 'Prediction error identification methods for stationary stochastic processes', *IEEE Trans.*, 1976, **AC - 21** (4), pp. 500 - 505.

[52] ÅSTRÖM, K. J. : 'Maximum likelihood and prediction error methods', *Automatica*, 1980, **16** (5), pp. 551 - 574.

[53] HAGENBLAD, A. : 'Initialization and model reduction for Wiener model identification', Technical Report No. LiTH-ISY-R-2150, University of Linköping, 1999.

[54] WESTWICK, D. and VERHAEGEN, M. : 'Identifying MIMO Wiener systems using subspace model identification methods', *Signal Processing*, 1996, **52** (2), pp. 235 - 258.

[55] HUNTER, I. W. and KORENBERG, M. J. : 'The identification of nonlinear biological systems : Wiener and Hammerstein cascade models', *Biological Cybernetics*, 1986, **55**, pp. 135 - 144.

[56] KALAFATIS, A. D., WANG, L. and CLUETT, W. R. : 'Identification of Wiener-type nonlinear systems in a noisy environment', *Int. J. Control*, 1997, **66** (6), pp. 923 - 941.

[57] VOROS, J. : 'Parameter identification of Wiener systems with discontinuous nonlinearities', *Systems and Control Letters*, 2001, **44** (5), pp. 363 - 372.

[58] BLOEMEN, H. H. J., CHOU, C. T., VAN DEN BOOM, T. J. J., VERDULT, V., VERHAEGEN, M. and BACKX, T. C. : 'Wiener model identification and predictive control for dual composition control of a distillation column', *Journal of Process Control*, 2001, **11** (6), pp. 601 - 620.

[59] KHAN, A. A. and VYAS, N. S. : 'Nonlinear bearing stiffness parameter estimation in flexible rotor-bearing systems using Volterra and Wiener approach', *Probabilistic Engineering Mechanics*, 2001, **16** (2), pp. 137 - 157.

[60] NORDSJO, A. E. and ZETTERBERG, L. H. : 'Identification of certain time-varying nonlinear Wiener and Hammerstein systems', *IEEE Trans.*, 2001, **SP - 49** (3), pp. 577 - 592.

[61] COLEMAN, T., BRANCH, M. A. and GRACE, A. : 'Optimization Toolbox for use with MATLAB', (The MathWorks Inc., Natick, Mass., 1999).

[62] BARKER, H. A. and PRADISTHAYON, T. : 'High-order autocorrelation functions of pseudorandom signals based on *m* sequences', *IEE Proc.*, 1970, **117** (9), pp. 1857 - 1863.

[63] ADBY, P. R. and DEMPSTER, M. A. H. : 'Introduction to Optimization Methods', (Chapman and Hall, 1974).

[64] OSBORNE, M. R. : 'Some aspects of non-linear least squares calculations', *in* LOOTSMA, F. A. (Ed.) : 'Numerical Methods for Non-linear Optimization', (Academic Press, 1972), Chap. 11.

[65] 'The Student Edition of MATLAB', The MathWorks Inc., (Prentice Hall, USA, 1992).

[66] NELDER, J. A. and MEAD, R. : 'A simplex method for function minimization', *Computer J.*, 1965, **7**, pp. 308 - 313.

[67] AKITT, J. W. : 'Function minimisation using the Nelder and Mead simplex method with limited arithmetic precision : the self regenerative simplex', *Computer J.*, 1977, **20** (1), pp. 84 - 85.

[68] OLSSON, D. M. and NELSON, L. S. : 'Nelder-Mead simplex procedure for function minimization', *Technometrics*, 1975, **17** (1), pp. 45 - 51.

[69] PARKINSON, J. M. and HUTCHINSON, D. : 'An investigation into the efficiency of variants on the simplex method', *in* LOOTSMA, F. A. (Ed.) : 'Numerical Methods for Non-linear Optimization', (Academic Press, 1972), Chap. 8.

[70] BARKER, H. A. and GODFREY, K. R. : 'System identification with multi-level periodic perturbation signals', *Control Engineering Practice*, 1999, **7** (6), pp. 717 - 726.

[71] WILLIAMS, R. J. and ZIPSER, D. : 'A learning algorithm for continually running, fully recurrent neural networks', *Neural Computat.*, 1989, pp. 1270 - 1289.

[72] WERBOS, P. J. : 'Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences', PhD. Dissertation, Harvard University, 1974.

[73] PARKER, D. B. : 'Learning logic', Technical Report No. TR-47, Massachusetts Institute of Technology, 1985.

[74] RUMELHART, D. E., HINTON, G. E. and WILLIAMS, R. J. : 'Learning internal representations by error propagation', *in* RUMELHART, D. E. and McCLELLAND, J. L. (Ed.) : 'Parallel Distributed Processing : Explorations in the Microstructure of Cognition', *Vol. 1 Foundations*, (Bradford Books/MIT Press, Cambridge, MA, 1986).

[75] DEMUTH, H. and BEALE, M. : 'Neural Network Toolbox for use with MATLAB', (The MathWorks Inc., Natick, Mass., 2001).

[76] GARDNER, J. W. and BARTLETT, P. N. : 'Electronic Noses', (Oxford University Press, Oxford, 1998).

[77] 'Electronic Noses and Olfaction 2000', GARDNER, J. W. and PERSAUD, K. C. (Ed.), (Institute of Physics Publishing, Bristol and Philadelphia, 2000).

[78] GARDNER, J. W. and BARTLETT, P. N. : 'A brief history of electronic noses', *Sensors and Actuators B*, 1994, **18 - 19** (1 - 3), pp. 211 - 220.

[79] GARDNER, J. W., SHURMER, H. V. and TAN, T. T. : 'Application of an electronic nose to the discrimination of coffees', *Sensors and Actuators B*, 1992, **6** (1 - 3), pp. 71 - 75.

[80] 'LabVIEW for Windows - User Manual', National Instruments, (National Instruments Corporation, Austin, 1993).

[81] 'LabVIEW - Function Reference Manual', National Instruments, (National Instruments Corporation, Austin, 1993).

[82] TAN, K. K., WANG Q. C., HANG, C. C. and HÄGGLUND, T. : 'Advances in PID Control', (Springer-Verlag London Limited, 1999).

[83] YU, C. C. : 'Autotuning of PID Controllers', (Springer-Verlag London Limited, 1999).

[84] ÅSTRÖM, K. J. and HÄGGLUND, T. : 'Automatic Tuning of PID Controllers', (Instrument Society of America, 1988).

[85] DORF, R. C. and BISHOP, R. H. : 'Modern Control Systems', 7th edition, (Addison-Wesley, 1995).

[86] JACQUOT, R. G. : 'Modern Digital Control Systems', (Marcel Dekker, 1981).

[87] PHILLIPS, C. L. and HARBOR, R. D. : 'Feedback Control Systems', 3rd edition, (Prentice Hall, USA, 1996).

[88] ÅSTRÖM, K. J. and HÄGGLUND, T. : 'The future of PID control', *Control Engineering Practice*, 2001, **9** (11), pp. 1163 - 1175.

[89] HÄGGLUND, T. and TENGVALL, A. : 'An automatic tuning procedure for unsymmetrical processes', *Proc. European Control Conference*, Rome, Italy, 1995, pp. 2450 - 2455.

[90] ZIEGLER, J. G. and NICHOLS, N. B. : 'Optimum settings for automatic controllers', *ASME Trans.*, 1942, **64**, pp. 759 - 768.

[91] ATHERTON, D. P. and KAYA, I. : 'Parameter estimation from relay autotuning with asymmetric limit cycle data', *Journal of Process Control*, 2001, **11** (4), pp. 429 - 439.

[92] RUBENSSON, M. and LENNERTSON B. : 'Stability of limit cycles in hybrid systems using discrete-time Lyapunov techniques', *Proc. IEEE Conference on Decision and Control*, **2**, Sydney, Australia, 2000, pp. 1397 - 1402.

[93] FRAMPTON, K. D. and CLARK. R. L. : 'Experiments on control of limit-cycle oscillations in a typical section', *Journal of Guidance, Control and Dynamics*, 2000, **23** (5), pp. 956 - 960.

[94] OPPENHEIM, A. V. and SCHAFER, R. W. : 'Digital Signal Processing', (Prentice Hall, USA, 1975).

[95] OPPENHEIM, A. V. and WILLSKY, A. S. : 'Signals and Systems', (Prentice Hall, USA, 1983).

[96] DeFATTA, D. J., LUCAS, J. G. and HOGDKISS, W. S. : 'Digital Signal Processing : A System Design Approach', (John Wiley & Sons, 1988).

[97] WANG, Q. G., BI, Q. and ZOU, B. : 'Use of FFT in relay feedback systems', *Electronics Letters*, 1997, **33** (12), pp. 1099 - 1100.

[98] LEE, T. H., WANG, Q. C. and TAN, K. K. : 'Knowledge-based process identification from relay feedback', *Journal of Process Control*, 1995, **5** (6), pp. 387 - 397.

[99] LEE, T. H., WANG, Q. C. and TAN, K. K. : 'A modified relay-based technique for improved critical point estimation in process control', *IEEE Trans.*, 1995, **CTS-3** (3), pp. 330 - 337.

[100] McCORMACK, A. S. and GODFREY, K. R. : 'Rule-based autotuning based on frequency domain estimation', *IEEE Trans.*, 1998, **CTS-6** (1), pp. 43 - 61.

[101] WELLSTEAD, P. E. : 'Nonparametric methods of system identification', *Automatica*, 1981, **17** (1), pp. 55 - 69.

[102] FOX, P. D. and GODFREY, K. R. : 'Multiharmonic perturbations for nonparametric autotuning', *IEE Proc. – Control Theory Appl.*, 1999, **146** (1), pp. 1 - 8.

[103] VODA, A. and LANDAU, I. D. : 'A method for the auto-calibration of PID controllers', *Automatica,* 1995, **31** (1), pp. 41 - 53.

[104] SHINSKEY, F. G. : 'PID-deadtime control of distributed processes', *Control Engineering Practice*, 2001, **9** (11), pp. 1177 - 1183.

[105] INGIMUNDARSON, A. and HÄGGLUND, T. : 'Robust tuning procedures of dead-time compensating controllers', *Control Engineering Practice*, 2001, **9** (11), pp. 1195 - 1208.

[106] DOEBELIN, E. O. : 'Control System Principles and Design', (John Wiley & Sons, 1985).

[107] PALMOR, Z. J. and BLAU, M. : 'An auto-tuner for Smith dead time compensator', *Int. J. Control*, 1994, **60** (1), pp. 117 - 135.

[108] LILIA, M. : 'Least squares fitting to a rational transfer function with time delay', *IEE Control Conference*, 1998, pp. 143 - 146.

[109] SUNG, S. W., LEE, I. B. and LEE, J. : 'New process identification method for automatic design of PID controllers', *Automatica*, 1998, **34** (4), pp. 513 - 520.

[110] PARK, J. H., PARK, H. I. and LEE, I. B. : 'Closed-loop on-line process identification using a proportional controller', *Chemical Engineering Science*, 1998, **53** (9), pp. 1713 - 1724.

[111] SUNG, S. W. and LEE, I. B. : 'On-line process identification and PID controller autotuning', *Korean Journal of Chemical* Engineering, 1999, **16** (1), pp. 45 - 55.

[112] CHERES, E. and EYDELZON, A. : 'Parameter estimation of a second order model in the frequency domain from closed loop data', *Transactions of the Institute of Chemical Engineers*, 2000, **78** (2), Part A, pp. 293 - 298.

[113] O'Dwyer, A. : 'Time delayed process model parameter estimation: a classification of techniques', *Proc. UKACC Control 2000*, Cambridge, U. K., 2000.

[114] SMITH, O. T. M. : 'Closed loop control of loops with dead time', *Chemical Engineering Progress*, 1957, **53** (5), pp. 217 - 219.

[115] HÄGGLUND, T. : 'A predictive PI controller for processes with long dead time', *IEEE Contr. Syst.*, February 1992, pp. 57 - 60.

[116] SHINSKEY, F. G. : 'How good are our controllers in absolute performance and robustness?', *Measurement and Control*, 1990, **23** (5), pp. 114 - 121.

[117] ÅSTRÖM, K. J., HÄGGLUND, T., HANG, C. C. and HO, W. K. : 'Automatic tuning and adaptation for PID controllers - a survey', *Control Engineering Practice*, 1993, **1** (4), pp. 699 - 714.

[118] MANN, G. K. I., HU, B. G. and GOSINE, R. G. : 'Time-domain based design and analysis of new PID tuning rules', *IEE Proc. - Control Theory Appl.*, 2001, **148** (3), pp. 251 - 261.

[119] DAHLIN, E. B. : 'Designing and tuning digital controllers', *Instruments and Control Systems*, 1968, **41** (6), pp. 77 - 83.

[120] ZHANG, W. D., SUN, Y. X. and XU, X. M. : 'Robust digital controller design for processes with dead times: new results', *IEE Proc. – Control Theory Appl.*, 1998, **145** (2), pp. 159 - 164.

[121] DE PAOR, A. M. : 'A modified Smith predictor and controller for unstable processes with time delay', *Int. J. Control*, 1985, **41** (4), pp. 1025 - 1036.

[122] DE PAOR, A. M. and EGAN, R. P. K. : 'Extension and partial optimisation of a modified Smith predictor and controller for unstable processes with time delay', *Int. J. Control*, 1989, **50** (4), pp. 1315 - 1326.

[123] MAJHI, S. and ATHERTON, D. P. : 'Modified Smith predictor and controller for processes with time delay', *IEE Proc. – Control Theory Appl.*, 1999, **146** (5), pp. 359 - 366.

[124] MAJHI, S. and ATHERTON, D. P. : 'Obtaining controller parameters for a new Smith predictor using autotuning', *Automatica*, 2000, **36** (11), pp. 1651 - 1658.

[125] HANG, C. C., WANG, Q. G. and CAO, L. S. : 'Self-tuning Smith predictors for processes with long dead time', *International Journal of Adaptive Control and Signal Processing*, 1995, **9** (3), pp. 255 - 270.

[126] PALMOR, Z. J. : 'Stability properties of Smith dead-time compensator controllers', *Int. J. Control*, 1980, **32** (6), pp. 937 - 949.

[127] 'SIMULINK - Dynamic System Simulation for MATLAB', (The MathWorks Inc., Natick, Mass., 2000).

[128] GERRY, J. P. and HANSEN, P. D. : 'Choosing the right controller', *Chem. Eng.*, May 25, 1987, **94** (8), pp. 65 - 68.

[129] PALMOR, Z. J. : 'Properties of optimal stochastic control systems with dead-time', *Automatica*, 1982, **18** (1), pp. 107 - 116.

[130] HANG, C. C., LEE, T. H. and HO. W. K. : 'Adaptive Control', (Instrument Society of America, Research Triangle Park, North Carolina, 1993).

[131] ÅSTRÖM, K. J., HANG, C. C., PERSSON, P. and HO, W. K. : 'Towards intelligent PID control', *Automatica*, 1991, **28** (1), pp. 1 - 9.

[132] HANG, C. C., ÅSTRÖM, K. J., and HO, W. K. : 'Refinements of the Ziegler-Nichols tuning formula', *IEE Proc. D*, 1991, **138** (2), pp. 111 - 118.

[133] 'Process Trainer PT 326', (Feedback Instruments Limited, Sussex, England, 1985).

[134] CHOW, C. M. : 'Design and Implementation of Digital Controllers', Third Year Project Report, University of Warwick, 1992.

[135] 'Real-Time Windows Target for use with Real-Time Workshop', (The MathWorks Inc., Natick, Mass., 2000).

[136] 'DAQ - PCI E Series User Manual', (National Instruments Corporation, 1997).

[137] 'Real-Time Workshop for use with Simulink', (The MathWorks Inc., Natick, Mass., 2000).

[138] OTTO, P. and WERNSTEDT, J. : 'Development and application of suboptimal test signal sequences for identifying nonlinear systems', *Proc. 7th IFAC Congress*, Helsinki, Finland, 1978, pp. 1927 - 1933.

[139] NARENDRA, K. S. and GALLMAN, P. G. : 'An iterative method for the identification of nonlinear systems using the Hammerstein model', *IEEE Trans.*, 1966, **AC - 11**, pp. 546 - 550.

[140] EVANS, C. , REES, D., JONES, L. and WEISS, M. : 'Periodic signals for measuring nonlinear Volterra kernels', *IEEE Trans.*, 1996, **IM - 45** (2), pp. 362 - 371.

[141] WEISS, M., EVANS, C. and REES, D. : 'Identification of nonlinear cascade systems using paired multisine signals', *IEEE Trans.*, 1998, **IM - 47** (1), pp. 332 - 336.

[142] VANDERSTEEN, G., ROLAIN, Y. and SCHOUKENS, J. : 'Non-parametric estimation of the frequency-response functions of the linear blocks of a Wiener-Hammerstein model', *Automatica*, 1997, **33** (7), pp. 1351 - 1355.

[143] NÉMETH, J. G., ISTVÁN, K. and SCHOUKENS, J. : 'Identification of Volterra kernels using interpolation', *Proc. IEEE Instrumentation and Measurement Technology Conference*, Budapest, Hungary, 2001, pp. 810 - 815.

[144] BERSHAD, N. J., BOUCHIRED, S. and CASTANIE, F. : 'Stochastic analysis of adaptive gradient identification of Wiener-Hammerstein systems for Gaussian inputs', *IEEE Trans.*, 2000, **SP - 48** (2), pp. 557 - 560.

[145] BAI, E. W. : 'An optimal two-stage identification algorithm for Hammerstein-Wiener nonlinear systems', *Automatica*, 1998, **34** (3), pp. 333 - 338.

[146] PINTELON, R. and SCHOUKENS, J. : 'System Identification - A Frequency Domain Approach', (IEEE Press, New York, 2001).

[147] FAKHOURI, S. Y. : 'Identification of Volterra kernels of nonlinear systems', *IEE Proc. D*, 1980, **127** (6), pp. 296 - 304.

[148] WEISS, M., EVANS, C. , REES, D. and JONES, L. : 'Structure identification of block-oriented nonlinear systems using periodic test signal', *Proc. IEEE Instrumentation and Measurement Technology Conference*, Brussels, Belgium, 1996, pp. 8 - 13.

[149] EVANS, C. , REES, D. and JONES, L. : 'Nonlinear disturbance errors in system identification using multisine test signals', *IEEE Trans.*, 1994, **IM - 43** (2), pp. 238 - 244.

[150] EVANS, C., REES, D. and JONES, L. : 'Identifying linear models of systems suffering nonlinear distortions, with a gas turbine application', *IEE Proc. - Control Theory Appl.*, 1995, **142** (3), pp. 229 - 240.

[151] SCHOUKENS, J. and PINTELON, R. : 'Identification of Linear Systems', (Pergamon Press, 1991).

[152] EVANS, C., REES, D. and HILL, D. : 'Frequency-domain identification of gas turbine dynamics', *IEEE Trans.*, 1998, **CST - 6** (5), pp. 651 - 662.

[153] EVANS, C., REES, D. and BORRELL, A. : 'Identification of aircraft gas turbine dynamics using frequency-domain techniques', *Control Engineering Practice*, 2000, **8** (4), pp. 457 - 467.

[154] AKAIKE, H. : 'Information theory and an extension of the maximum likelihood principle', *Proc. 2nd International Symposium on Information Theory*, 1972, pp. 267 - 281.

[155] AKAIKE, H. : 'Modern development of statistical methods', *in* EYKHOFF, P. (Ed.) : 'Trends and Progress in System Identification', (Pergamon Press, Oxford, 1981).

[156] BHANSALI, R. J. and DOWNHAM, D. Y. : 'Some properties of the order of an autoregressive model selected by generalization of Akaike's FPE criterion', *Biometrica*, 1977, **64**, pp. 547 - 551.

[157] SCHOUKENS, J. and PINTELON, R. : 'Identification of linear systems in the presence of model errors', *Proc. IEEE Conference on Instrumentation and Measurement Technology*, Hamamatsu, Japan, 1994, pp. 1372 - 1376.

[158] KRAUSS, T. P., SHURE, L. and LITTLE, J. N. : 'Signal Processing Toolbox for use with MATLAB', (The MathWorks Inc., Natick, Mass., 1995).

[159] EVANS, C. and REES, D. : 'Nonlinear distortions and multisine signals - part I : measuring the best linear approximation', *IEEE Trans.*, 2000, **IM - 49** (3), pp. 602 - 609.

[160] EVANS, C. and REES, D. : 'Nonlinear distortions and multisine signals - part II : minimizing the distortion', *IEEE Trans.*, 2000, **IM - 49** (3), pp. 610 - 616.

[161] HABER, R. and UNBEHAUEN, H. : 'Structure identification of nonlinear dynamic systems - a survey on input/output approaches', *Automatica*, 1990, **26** (4), pp. 651 - 677.

[162] LAWRENCE, P. J. : 'Estimation of the Volterra functional series of a nonlinear system using frequency response data', *IEE Proc. D*, 1981, **128** (5), pp. 206 - 210.

[163] IM, S. and POWERS, E. J. : 'Sparse third-order orthogonal frequency-domain Volterra-like model', *Journal of the Franklin Institute*, 1996, **333B** (3), pp. 385 - 412.

[164] POWERS, E. J., IM, S., KIM, S. B. and TSENG, C. H. : 'Frequency-domain Volterra kernel estimation via higher-order statistical signal processing', *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, 1993, **1**, pp. 445 - 450.

Appendix

Script File : ***prs_perturbation***

```
% Programme to Generate Pseudo-random Perturbation Signals
% ******************************************************
%
% This programme allows the user to generate pseudo-random perturbation signals.
% These include binary and ternary signals.
% Two types of harmonic specifications are available:
%   1. All harmonics present
%   2. Even harmonics suppressed
%
% Author : Ai Hui Tan
% Date last modified : 2/11/2001

continue_harmonics = 'y';
while continue_harmonics =='y'
  choice = menu ('Choose a harmonic specification:','signals with no harmonics
suppressed','signals with even harmonics suppressed');

  if choice == 1
%all harmonics present
    prs_all_harmonics
  elseif choice == 2
%even harmonics suppressed
    prs_odd_harmonics
  end

  continue_harmonics = 'a';
  while (continue_harmonics~='y') & (continue_harmonics~='n')
    continue_harmonics = input('Do you want a signal with a different harmonic specification?
y/n [y]:','s');
    if isempty(continue_harmonics)
      continue_harmonics = 'y';
    end
  end
end
```

Function files : These are arranged in alphabetical order.

function Rxx = ***autocorrelation***(bitseries)

```
%Calculates the periodic autocorrelation for a given signal.
%
%   Rxx = autocorrelation(bitseries)
%
%   Input argument:
%   bitseries = Vector containing the amplitudes of the signal sequence
%
```

```
%   Output argument:
%   Rxx = Vector of the autocorrelation amplitude
%
%   Algorithm:
%     For the signal x(r) of length N with discrete Fourier transform X(k),
%     the autocorrelation is calculated from the formula
%
%                                N-1
%     autocorrelation(r) = sum X(k)X(k)' exp(2*pi*k*r/N)
%                                k=0
%
%     where X(k)' is the complex conjugate of X(k).
%
%     The resulting autocorrelation is then normalised by dividing it by N.
%
%   Example:
%     bitseries = [1 1 -1 1 -1 1];
%     Rxx = autocorrelation(bitseries);

vector = (ifft(abs(fft(bitseries)).^2))/length(bitseries);
Rxx = real(vector);


function autocorrelation_plot(bitseries,full_period,key)

%Calculates the periodic autocorrelation magnitude and plots it against delay/clock-pulse
%interval. Allows the user to obtain the values of the autocorrelation magnitude by entering the
%delay/clock-pulse interval at the command window.
%
%   autocorrelation_plot(bitseries,full_period,key)
%
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   full_period = Vector containing the clock pulses plus one extra pulse
%   key = Parameter specifying the harmonic content of the signal
%       This can take only the values 1 or 2.
%       1 : For use with pseudo-random signals with all harmonics present.
%           This will not activate a 'cursor' in the command window.
%           The autocorrelation magnitude will be printed on the screen.
%       2 : For use with pseudo-random signals with some harmonics suppressed and other
%           signals such as computer-generated signals.
%           Activates a 'cursor' in the command window.
%           Entering -1 for delay/clock-pulse interval will terminate this 'cursor' function.
%
%   Example:
%     bitseries = [1; 1; -1; 1; -1; 1];
%     full_period = [1:7];
%     autocorrelation_plot(bitseries,full_period,2)

%calculates the autocorrelation magnitude
Rxx = autocorrelation(bitseries);
full_Rxx = [Rxx; Rxx(1)];
%plots graph of autocorrelation magnitude against delay/sampling interval
```

```
figure(4)
plot(full_period-1,full_Rxx)
title ('Plot of Autocorrelation Function of Signal')
axis([0 length(full_period)-1 min(Rxx)-1/length(bitseries) max(Rxx)+1/length(bitseries)])
xlabel ('Delay/Clock-pulse Interval')
ylabel ('Autocorrelation')
zoom on

if mod(length(bitseries),2)==1 & (key==1)
%prints the autocorrelation magnitude
  fprintf('Autocorrelation value with no delay is %f.\n', Rxx(1))
  fprintf('Autocorrelation value with %d to %d units of delay is %f.\n\n', 1, length(bitseries)-1,
Rxx(2))
else
%activates 'cursor'
  disp('Enter the delay/clock-pulse interval to see its autocorrelation magnitude. If you do not
want this function, enter -1.')
  delay = 0;

  while (delay~=-1)
    delay = input('Delay/clock-pulse interval = ');
    while (delay<0) & (delay~=-1) | (delay>length(bitseries)-1) | (mod(delay,1)~=0)
      fprintf('Please enter an integer between 0 and %d or enter -1 to quit.\n', length(bitseries)-1)
      delay = input('Delay/clock-pulse interval = ');
    end
    if (delay~=-1)
      fprintf('The autocorrelation magnitude is %f.\n\n', Rxx(delay+1))
    end
  end
  fprintf('\n')
end


function EMINE = calculate_EMINE(bitseries,freqv)

%Calculates the minimum ratio of the actual harmonic amplitude and the specified harmonic
%amplitude for a signal with a flat spectrum specified.
%
%   EMINE = calculate_EMINE(bitseries,freqv)
%
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   freqv = Vector of the specified non-zero frequencies
%
%   Output argument:
%   EMINE = minimum ratio of the actual to the specified harmonic amplitude
%
%   Algorithm:
%   |C'u(k)| = sqrt(2)*|sin(pi*k/N)/(pi*k)*U(k)
%   where
%     |C'u(k)| = actual harmonic amplitude at harmonic number k
%     U(k) = discrete Fourier transform of the discrete signal
%     N = signal length
```

```
%                                                R
%   EMINE = 100*minimum( |C'u(k)| / sqrt (sum (|C'u(k)|^2) /R) ) %
%                    k=1,2,..,R                  k=1
%   where R < N/2
%
%   Example:
%   bitseries = [1 -1 1 -1 -1 1 -1 1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1];
%   freqv = 1:9;
%   EMINE = calculate_EMINE(bitseries,freqv)

DFT = abs(fft(bitseries));
%calculates actual harmonic amplitudes
for k = 1:length(freqv)
  angle(k) = pi*(freqv(k))/length(bitseries);
  amp(k) = sqrt(2)*abs(sin(angle(k)).*DFT(freqv(k)+1))/angle(k)/length(bitseries);
end
%calculates the minimum ratio of the actual to the specified harmonic amplitude
EMINE = 100*min(amp)/(sqrt(sum(amp.^2)/length(freqv)));


function PIPS = calculate_PIPS(bitseries)

%Calculates the performance index for perturbation signals, PIPS.
%
%   PIPS = calculate_PIPS(bitseries)
%
%   Input argument:
%   bitseries = Vector containing the amplitudes of the signal sequence
%
%   Output argument:
%   PIPS = performance index for perturbation signals
%
%   Algorithm:
%                          N-1
%   PIPS = 200 * sqrt (sum (|U(k)|^2) ) / N / (umax-umin) %
%                          k=1
%   where
%     U(k) = discrete Fourier transform of the discrete signal
%     N = signal length
%     k = harmonic number
%     umax = maximum value of the signal
%     umin = minimum value of the signal
%
%   Example:
%   bitseries = [1 -1 1 -1 -1 1 -1 1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1];
%   PIPS = calculate_PIPS(bitseries);

DFT = abs(fft(bitseries));
for k = 2:length(bitseries)
  DFT_squared(k) = DFT(k)*DFT(k);
end
PIPS = 200*sqrt(sum(DFT_squared))/length(bitseries)/(max(bitseries)-min(bitseries));
```

```
function PIPSE = calculate_PIPSE(bitseries,freqv)

%Calculates the effective performance index for perturbation signals, PIPSE.
%
%   PIPSE = calculate_PIPSE(bitseries,freqv)
%
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   freqv = Vector containing the non-zero harmonics
%
%   Output argument:
%   PIPSE = effective performance index for perturbation signals
%   Algorithm:
%                              R
%   PIPSE = 200 * sqrt (sum (|Cu'(k)|^2) ) / (umax-umin) %
%                             k=1
%   where
%     |C'u(k)| = sqrt(2)*|sin(pi*k/N)/(pi*k)*U(k)
%     R < length/2
%     umax = maximum value of the signal
%     umin = minimum value of the signal
%
%   Example:
%   bitseries = [1 -1 1 -1 -1 1 -1 1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1];
%   freqv = 1:9;
%   PIPSE = calculate_PIPSE(bitseries,freqv);

DFT = abs(fft(bitseries));

%calculates actual power
power = 0;
for k = 1:length(freqv)
  if freqv(k) == 0
    amp(k) = DFT(1)/length(bitseries);
    power(k) = amp(k)*amp(k);
  else
    angle(k) = pi*(freqv(k))/length(bitseries);
    amp(k) = abs(sin(angle(k)).*DFT(freqv(k)+1)/angle(k)/length(bitseries));
    power(k) = 2*amp(k)*amp(k);
  end
end

%calculates effective performance index
PIPSE = 200*sqrt(sum(power))/(max(bitseries)-min(bitseries));


function R0 = calculate_R0(x)

%Calculates the upper bound in the crosscorrelation without unsystematic errors due to second-
%order nonlinearities.
%
%   R0 = calculate_R0(x)
%
```

```
%   Input argument:
%   x = Input sequence
%
%   Output argument:
%   R0 = Upper bound in the crosscorrelation without second-order unsystematic errors
%
%   Example:
%     x = mlbs(7);
%     R0 = calculate_R0(x)

a=length(x);
for k = 1:a
  if x(k) == -1
    x(k) = 0;
  end
end
x = [x;x];store = a;

for m=0:a-1
  if (m >= store)
    break
  end
  for n=m+1:a-1
    if (n >= store)
      break
    end
    for p=n+1:a-1
      if (p >= store)
        break
      end
      if mod(x([a:2*a]-m)+x([a:2*a]-n)+x([a:2*a]-p),2)==0
        if p < store
          store = p;
        end
      end
    end
  end
end
R0 = store-1;


function R1 = calculate_R1(x)

%Calculates the upper bound in the crosscorrelation without unsystematic errors due to third-
%order nonlinearities.
%
%   R1 = calculate_R1(x)
%
%   Input argument:
%   x = Input sequence
%
%   Output argument:
%   R1 = Upper bound in the crosscorrelation without third-order unsystematic errors
```

```
%   Example:
%      x = mlbs(7);
%      R1 = calculate_R1(x)

a=length(x);
for k = 1:a
  if x(k) == -1
    x(k) = 0;
  end
end

x = [x;x];store = a;

for m=0:a-1
  if (m >= store)
    break
  end
  for n=m+1:a-1
    if (n >= store)
      break
    end
    for p=n+1:a-1
      if (p >= store)
        break
      end
      for q=p+1:a-1
        if (q >= store)
          break
        end
        if mod(x([a:2*a]-m)+x([a:2*a]-n)+x([a:2*a]-p)+x([a:2*a]-q),2)==0
          if q < store
            store = q;
          end
        end
      end
    end
  end
end

R1 = store-1;


function DFT_plot(bitseries,clock_pulse,key)

%Calculates the discrete Fourier Transform magnitude and plots it against harmonic number.
%Allows the user to obtain the values of the discrete Fourier transform magnitude by entering
%the harmonic number at the command window.
%
%   DFT_plot(bitseries,clock_pulse,key)
%
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   clock_pulse = Vector containing the clock pulses
```

```
%   key = Parameter specifying the harmonic content of the signal
%         This can take only the values 1 or 2.
%         1 : For use with pseudo-random signals.
%             This will not activate a 'cursor' in the command window.
%             The discrete Fourier transform magnitude will be printed on the screen.
%         2 : For use with other signals such as computer-generated signals.
%             Activates a 'cursor' in the command window.
%             Entering -1 for harmonic number will terminate this 'cursor' function.
%
%   Example:
%     bitseries = [1 1 -1 1 -1 1];
%     clock_pulse = [1:6];
%     DFT_plot(bitseries,clock_pulse,2)

DFT = abs(fft(bitseries));

%plots graph of discrete Fourier transform magnitude against harmonic number
figure(2)
stem(clock_pulse-1,DFT,'-')
title ('Plot of Discrete Fourier Transform of Discrete Signal')
axis([0  length(DFT)-1  0  max(DFT)])
xlabel ('Harmonic Number')
ylabel ('Discrete Fourier Transform Magnitude')
zoom on

%prints the discrete Fourier transform magnitude if key is 1 and activates 'cursor' if key is 2.
if key == 1
  if mod(length(bitseries),2)==1
    fprintf('DFT magnitude at dc is %f.\n', DFT(1))
    fprintf('DFT magnitude at other harmonics within the period is %f.\n', DFT(2))
  else
    fprintf('DFT magnitude at odd harmonics within a period, except at harmonic %d is %f.\n',
length(bitseries)/2, DFT(2))
    fprintf('DFT magnitude at harmonic %d is %f.\n', length(bitseries)/2,
DFT((length(bitseries)/2)+1))
  end

else
  disp('Enter a harmonic number to see its DFT magnitude. If you do not want this function,
enter -1.')
  number = 0;

  while (number~=-1)
    number = input('Harmonic number = ');
    while (number<0) & (number~=-1) | (number>length(bitseries)-1) | (mod(number,1)~=0)
      fprintf('Please enter an integer between 0 and %d or enter -1 to quit.\n',length(bitseries)-1)
      number = input('Harmonic number = ');
    end
    if (number~=-1)
      fprintf('The DFT magnitude is %f.\n\n', DFT(number+1))
    end
  end
end
```

```
function HAB_series = HAB(length)

%Calculates the Hall series given the signal length.
%
%   HAB_series = HAB(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   HAB_series = Amplitude vector of the generated Hall series
%   Algorithm:
%     A primitive root u is first found.
%     The element r of the signal, a(r) is 1 if r=u^t(mod(length))
%     where t = 0, 1 or 3 (mod 6).
%     a(r) is -1 otherwise.
%
%   Example:
%     HAB_series = HAB(31);

%sets the primitive root
if length == 1051
  root = 7;
elseif (length == 1471) | (length == 16927)
  root = 6;
else
  root = 3;
end

HAB_series = -ones(length,1);
%calculates Hall series
HAB_series(1) = 1;HAB_series(root) = 1;
power = 3;remainder = root^power;position = 'a';
HAB_series(remainder) = 1;

while power<=length-1
  if position == 'a';
    added_power = 3;
    position = 'b';
  elseif position == 'b';
    added_power = 1;
    position = 'c';
  else position == 'c';
    added_power = 2;
    position = 'a';
  end

  power = power + added_power;
  if power <=length-1
    remainder = mod(remainder*(root^added_power),length);
    HAB_series(remainder) = 1;
  end
end
```

```
function exist = HAB_exist(length)

%Checks if the Hall series exists for a particular length.
%
%   HAB_exist(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   exist = 1 means that the Hall series exists.
%         = 0 means that the Hall series does not exist.
%
%   Example:
%     exist = HAB_exist(46);

%Hall series cannot be of length smaller than 31
if length<31
  exist = 0;
else
  root = sqrt((length-27)/4);
  if (prime(length)==1) & (mod(root,1)==0)
    exist = 1;
  else
    exist = 0;
  end
end


function inverse = inverse_repeat(bitseries)

%Calculates the inverse-repeat of a given signal.
%
%   inverse = inverse_repeat(bitseries)
%
%   Input argument:
%   bitseries = Vector containing the amplitudes of the signal sequence
%
%   Example:
%   bitseries = [1; 1; -1; 1; -1; -1; 1];
%   inverse = inverse_repeat(bitseries);

%concatenates two periods of a given signal
inverse = [bitseries; bitseries];

%changes the sign of every other element of the signal
for number = 2:2:length(inverse)
  inverse(number) = -(inverse(number));
end
```

```
function MLB_series = MLB(length)

%Calculates the MLB sequence given the signal length.
%
%   MLB_series = MLB(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   MLB_series = Amplitude vector of the generated maximum length binary sequence
%
%   Example:
%     MLB_series = MLB(127);

%gives the user choice whether to specify characteristic polynomial
answer = 'a';
while (answer~='y') & (answer~='n')
  answer=input('Do you want to enter the characteristic polynomial? y/n [y]:','s');
  if isempty(answer);
    answer = 'y';
  end
end

%generates maximum length binary sequence
if answer == 'y'
  poly = input('Characteristic polynomial = ');poly_exist = poly_check(poly);
  while poly_exist == 0
    disp('The polynomial entered is not primitive and irreducible.')
    choice=input('Do you want to reenter the characteristic polynomial? y/n [y]:','s');
    if isempty(choice);
      choice = 'y';
    end
    if choice ~= 'y'
      break
    end
    poly = input('Characteristic polynomial = ');poly_exist = poly_check(poly);
  end
  if poly_exist == 1
    MLB_series = poly_MLB(poly);
  else
    answer = 'n';
  end
end

%generates default maximum length binary sequence if characteristic polynomial not given
if answer == 'n'
  disp('The default maximum length binary sequence will be generated.')
  disp('Please strike a key to continue.')
  pause
  n = round(log(length+1)/log(2));
  MLB_series = mlbs(n);
end
```

```
function exist = MLB_exist(length)

%Checks if the maximum length binary sequence exists for a particular length.
%
%   MLB_exist(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   exist = 1 means that the maximum length binary sequence exists.
%         = 0 means that the maximum length binary sequence does not exist.
%
%   Example:
%     exist = MLB_exist(46);

%calculates the integer above the signal length
number = length+1;

%checks if the integer above the signal length is a power of 2
while number>2
  number = number/2;
end
if number == 2
  exist=1;
else
  exist=0;
end


function poly_exist = poly_check(poly)

%Checks if a polynomial is primitive and irreducible.
%
%   poly_exist = poly_check(poly)
%
%   Input argument:
%   poly = Characteristic polynomial
%
%   Example:
%     poly_exist = poly_check([1 0 0 0 1 0 0 1])

%generates the polynomial
series = poly_MLB(poly);

%calculates the discrete Fourier transform and checks if it is flat
X = abs(fft(series));
X(1) = [];
if min(round(10000*X)) == max(round(10000*X))
  poly_exist = 1;
else
  poly_exist = 0;
end
```

```
function MLB_series = poly_MLB(poly)

%Calculates the maximum length binary sequence for a given characteristic polynomial.
%
%   MLB_series = poly_MLB(poly)
%
%   Input argument:
%   poly = Characteristic polynomial of the sequence
%
%   Output argument:
%   MLB_series = Amplitude vector of the generated maximum length binary sequence
%
%   Example:
%     MLB_series = poly_MLB([1 0 0 0 1 0 0 1]);

size_poly = size(poly);

%initializes states in shift register
x(1,1) = 1;
for n = 2:size_poly(2)-1
  x(1,n) = 0;
end
bitseries(1) = -1;

%calculates maximum length binary sequence
for t = 2:2^(size_poly(2)-1)-1
  x(t,1) = mod(sum(poly(2:size_poly(2)).*x(t-1,(1:size_poly(2)-1))),2);
  for n = 2:size_poly(2)-1
    x(t,n) = x(t-1,n-1);
    if x(t,size_poly(2)-1) == 1
      bitseries(t) = 1;
    else
      bitseries(t) = -1;
    end
  end
end
MLB_series = bitseries';


function power_plot(bitseries,harmonic)

%Calculates power and plots it against harmonic number. Allows the user to obtain the values
%of the power by entering the harmonic number at the command window. Entering -1 for
%harmonic number will terminate the 'cursor' function.
%
%   power_plot(bitseries,harmonic)
%
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   harmonic = Vector of three periods of clock pulses
%
```

```
%   Example:
%      bitseries = [1; 1; -1; 1; -1; 1];
%      harmonic = [1:18];
%      power_plot(bitseries,harmonic)

DFT = abs(fft(bitseries));
three_periods = [DFT; DFT; DFT];

%calculates power
amp(1) = DFT(1)/length(bitseries);
for k = 2:length(three_periods)
  angle(k) = pi*(k-1)/length(bitseries);
  amp(k) = sqrt(2)/length(bitseries)*abs(sin(angle(k))*three_periods(k)/angle(k));
  power(k) = amp(k)^2;
end
%plots power against harmonic number
figure(3)
stem(harmonic-1,power,'-')
title ('Plot of Power Spectrum Discrete Signal')
axis([0  length(power)-1  0  max(power)])
xlabel ('Harmonic Number')
ylabel ('Power Spectrum Magnitude')
zoom on

%provides 'cursor' function
disp('Enter a harmonic number to see its power spectrum magnitude. If you do not want this
function, enter -1.')
value = 0;
while (value~=-1)
  value = input('Harmonic number = ');
  while (value<0) & (value~=-1) | (value>3*length(bitseries)-1) | (mod(value,1)~=0)
    fprintf('Please enter an integer between 0 and %d or enter -1 to quit.\n', 3*length(bitseries) -
1);
    value = input('Harmonic number = ');
  end
  if (value~=-1)
    fprintf('The power is %f.\n\n', power(value+1))
  end
end


function y = prime(length)

%Checks if a given number is a prime.
%
%   y = prime(length)
%
%   Input argument:
%   length = Number to be checked if it is a prime
%
%   Output argument:
%   y = 1 if the number is a prime
%     = 0 if it is not a prime
```

```
%   Example:
%      y = prime(7);

y=1;

%integers 1 and 2 are considered primes
if (length==1) | (length==2)
  y = 1;
%the number is not a prime if it is even
elseif (mod(length,2)==0)
  y = 0;

%if the number is odd, divide it by odd numbers smaller than itself to check if
%it is prime
else
  for number = 3:2:length-1
    if (mod(length,number)==0)
      y=0;
      break
    end
  end
end
```

function ***prs_all_harmonics***

```
%Generates pseudo-random signals with all harmonics present.
%These include binary and ternary signals.
%
%Pseudo-random signals available are:
%   maximum length binary sequences
%   quadratic residue binary sequences
%   Twin Prime sequences
%   Hall sequences
%   quadratic residue ternary sequences

%allows the user the choice of signal length
repeat_length = 'y';
while repeat_length == 'y'

  length=input('Enter the length of the signal which can only be an odd integer between 3 and
49999 inclusive.\n');
  while (length<3) | (length>49999) | ((mod(length,2))~=1)
    length=input('Please enter an odd integer between 3 and 49999 inclusive.\n');
  end

%allows the user the choice of the number of signal levels
  repeat_level = 'y';
  while repeat_level == 'y'
    level = menu ('Choose one of the following:','binary signal','ternary signal');

%generates binary signals
    if level == 1
```

```
%checks if pseudo-random signals are available
      if (MLB_exist(length)==1) | (QRB_exist(length)==1) | (TPB_exist(length)==1) |
(HAB_exist(length)==1)
        count = 0;

%prints on the screen the classes of pseudo-random signals available
        disp ('Pseudo-random binary signal(s) of this length can be based on the following
classes:')

        if MLB_exist(length) == 1
          disp ('Maximum length binary signal')
          count = count+1;
          MLB_present = 1;
        else
          MLB_present = 0;
        end
        if QRB_exist(length) == 1
          disp ('Quadratic residue binary signal')
          count = count +1;
          QRB_present = 1;
        else
          QRB_present = 0;
        end
        if TPB_exist(length) == 1
          disp ('Twin Prime signal')
          count = count+1;
          TPB_present = 1;
        else
          TPB_present = 0;
        end
        if HAB_exist(length) == 1
          disp ('Hall signal')
          count = count+1;
          HAB_present = 1;
        else
          HAB_present = 0;
        end
        fprintf ('\n')

%gives user the choice to continue or abort
        continue = 'a';
        while (continue~='y') & (continue~='n')
          continue = input('Do you want to generate the signal? y/n [y]:','s');
          if isempty(continue)
            continue = 'y';
          end
        end

        if continue == 'y'
          if count>1
          repeat_class = 'y';
            while repeat_class == 'y'
```

```
%allows user the choice of signal class
                disp('Choose a signal class by entering the associated character.')
                if MLB_present == 1
                  disp ('Maximum length binary signal : m')
                end
                if QRB_present == 1
                  disp ('Quadratic residue binary signal : b')
                end
                if TPB_present == 1
                  disp ('Twin Prime signal : t')
                end
                if HAB_present == 1
                  disp ('Hall signal : h')
                end

                wrong = 1;
                while wrong == 1

                  class = input('','s');
                  fprintf ('\n')

%generates the pseudo-random signal class chosen
%generates maximum length binary signal
                  if class == 'm'
                    if MLB_present == 1
                      MLB_series = MLB(length);
                      signal_display (MLB_series)

%calculates upper bound in the crosscorrelation without unsystematic errors due to second and
%third order nonlinearities

                      R0 = calculate_R0(MLB_series);
                      R1 = calculate_R1(MLB_series);
                      fprintf('Upper bound in the crosscorrelation without unsystematic errors due
to second-order nonlinearities, R0 is %d.\n\n', R0)
                      fprintf('Upper bound in the crosscorrelation without unsystematic errors due
to third-order nonlinearities, R1 is %d.\n\n', R1)

%allows the user to calculate second-order terms
                      answer = 'a';
                      while (answer~='y') & (answer~='n')
                        answer=input('Do you wish to calculate second-order terms? y/n [y]:','s');
                        if isempty(answer);
                          answer = 'y';
                        end
                      end

                      if answer == 'y'
                        fprintf('Please enter an integer between 1 and %d or enter -1 to quit.\n',
length-1);
                        delay = 1;
                        while (delay~=-1)
                          delay = input('Delay = ');
```

```
                    while ((delay<1) | (delay>length-1)) & (delay~=-1)
                      fprintf('Please enter an integer between 1 and %d or enter -1 to quit.\n',
length-1);
                      delay = input('Delay = ');
                    end
                    if (delay~=-1)
                      shift = second_shift(MLB_series,delay);
                      fprintf('The resulting shift is %d.\n\n', shift)
                    end
                  end
                end

%allows the user to calculate third-order terms
                answer = 'a';
                while (answer~='y') & (answer~='n')
                  answer=input('Do you wish to calculate third-order terms? y/n [y]:','s');
                  if isempty(answer);
                    answer = 'y';
                  end
                end

                if answer == 'y'
                  fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to
quit.\n', length-1);
                  disp('The integers must be increasing in value and entered in the format
[integer1 integer2].')
                  delay1 = 1;
                  delay2 = 2;
                  while (delay1~=-1) | (delay2~=-1)
                    delay = input('Delays = ');
                    delay1 = delay(1);
                    delay2 = delay(2);
                    while ((delay1<1) | (delay1>length-1) | (delay2<1) | (delay2>length-1) |
(delay1>delay2)) & ((delay1~=-1) | (delay2~=-1))
                      fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to
quit.\n', length-1);
                      disp('The integers must be increasing in value and entered in the
format [integer1 integer2].')
                      delay = input('Delays = ');
                      delay1 = delay(1);
                      delay2 = delay(2);
                    end
                    if (delay1~=-1) & (delay2~=-1)
                      shift = third_shift(MLB_series,delay1,delay2);
                      fprintf('The resulting shift is %d.\n\n', shift)
                    end
                  end
                end

                wrong = 0;
              else
                wrong = 1;
              end
```

```
    %generates quadratic residue binary signal
                elseif class == 'b'
                  if QRB_present == 1
                    QRB_series = QRB(length);
                    signal_display (QRB_series)
                    wrong = 0;
                  else
                    wrong = 1;
                  end

%generates Twin Prime signal
                elseif class == 't'
                  if TPB_present == 1
                    TPB_series = TPB(length);
                    signal_display (TPB_series)
                    wrong = 0;
                  else
                    wrong = 1;
                  end
%generates Hall signal
                elseif class == 'h'
                  if HAB_present == 1
                    HAB_series = HAB(length);
                    signal_display (HAB_series)
                    wrong = 0;

                  else
                    wrong = 1;
                  end

                else
                  wrong = 1;
                end

                if wrong == 1
                  disp ('Please only enter an available choice.')
                end
              end

              repeat_class = 'a';
              while (repeat_class~='y') & (repeat_class~='n')
                repeat_class = input('Do you want to generate a pseudo-random signal based on
a different class? y/n [y]:','s');
                if isempty(repeat_class)
                  repeat_class = 'y';
                end
              end
            end
          else
%generates maximum length binary signal
            if MLB_present == 1
              MLB_series = MLB(length);
              signal_display (MLB_series)
```

```
%calculates upper bound in the crosscorrelation without unsystematic errors due to second and
%third order nonlinearities
            R0 = calculate_R0(MLB_series);
            R1 = calculate_R1(MLB_series);
            fprintf('Upper bound in the crosscorrelation without unsystematic errors due to
second-order nonlinearities, R0 is %d.\n\n', R0)
            fprintf('Upper bound in the crosscorrelation without unsystematic errors due to
third-order nonlinearities, R1 is %d.\n\n', R1)

%allows the user to calculate second-order terms
            answer = 'a';
            while (answer~='y') & (answer~='n')
              answer=input('Do you wish to calculate second-order terms? y/n [y]:','s');
              if isempty(answer);
                answer = 'y';
              end
            end
            if answer == 'y'
              fprintf('Please enter an integer between 1 and %d or enter -1 to quit.\n', length-
1);
              delay = 1;

              while (delay~=-1)
                delay = input('Delay = ');
                while ((delay<1) | (delay>length-1)) & (delay~=-1)
                  fprintf('Please enter an integer between 1 and %d or enter -1 to quit.\n',
length-1);
                  delay = input('Delay = ');
                end
                if (delay~=-1)
                  shift = second_shift(MLB_series,delay);
                  fprintf('The resulting shift is %d.\n\n', shift)
                end
              end
            end

%allows the user to calculate third-order terms
            answer = 'a';
            while (answer~='y') & (answer~='n')
              answer=input('Do you wish to calculate third-order terms? y/n [y]:','s');
              if isempty(answer);
                answer = 'y';
              end
            end

            if answer == 'y'
              fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to quit.\n',
length-1);
              disp('The integers must be increasing in value and entered in the format
[integer1 integer2].')
              delay1 = 1;
              delay2 = 2;
```

```
                while (delay1~=-1) | (delay2~=-1)
                  delay = input('Delays = ');
                  delay1 = delay(1);
                  delay2 = delay(2);
                  while ((delay1<1) | (delay1>length-1) | (delay2<1) | (delay2>length-1) |
(delay1>delay2)) & ((delay1~=-1) | (delay2~=-1))
                    fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to
quit.\n', length-1);
                    disp('The integers must be increasing in value and entered in the format
[integer1 integer2].')
                    delay = input('Delays = ');
                    delay1 = delay(1);
                    delay2 = delay(2);
                  end
                  if (delay1~=-1) & (delay2~=-1)
                    shift = third_shift(MLB_series,delay1,delay2);
                    fprintf('The resulting shift is %d.\n\n', shift)
                  end
                end
              end

%generates quadratic residue binary signal
            elseif QRB_present == 1
              QRB_series = QRB(length);
              signal_display (QRB_series)

%generates Twin Prime signal
            elseif TPB_present == 1
              TPB_series = TPB(length);
              signal_display (TPB_series)

%generates Hall signal
            elseif HAB_present == 1
              HAB_series = HAB(length);
              signal_display (HAB_series)
            end
          end
        end

%no pseudo-random binary signals available
      else
        disp ('No pseudo-random binary signal is available for this length.')
        fprintf ('\n')
      end
    end

%generates ternary signals
    if level == 2
      if (QRT_exist(length)==1)
        disp ('Pseudo-random ternary signal(s) of this length can be based on the following
classes:')
        disp ('Quadratic residue ternary signal')
        fprintf ('\n')
```

```
%gives user the choice to continue or abort
        continue = 'a';
        while (continue~='y') & (continue~='n')
          continue = input('Do you want to generate the signal? y/n [y]:','s');
          if isempty(continue)
            continue = 'y';
          end
        end

%generates quadratic residue ternary signal
        if continue == 'y';
          QRT_series = QRT(length);
          signal_display (QRT_series)
        end

%no pseudo-random ternary signals available
      else
        disp ('No pseudo-random ternary signal is available for this length.')
        fprintf ('\n')
      end
    end

    repeat_level = 'a';
    while (repeat_level~='y') & (repeat_level~='n')
      repeat_level = input('Do you want to generate a signal with a different number of levels?
y/n [y]:','s');
      if isempty(repeat_level)
        repeat_level = 'y';
      end
    end
  end

  repeat_length = 'a';
  while (repeat_length~='y') & (repeat_length~='n')
    repeat_length = input('Do you want to generate a signal with a different length? y/n [y]:','s');
    if isempty(repeat_length)
      repeat_length = 'y';
    end
  end
end
```

function ***prs_odd_harmonics***

```
%Generates pseudo-random signals with only odd harmonics present.
%These include binary and ternary signals.
%
%Pseudo-random signals available are:
%   maximum length binary inverse-repeat sequences
%   quadratic residue binary inverse-repeat sequences
%   Twin Prime inverse-repeat sequences
%   Hall inverse-repeat sequences
%   quadratic residue ternary inverse-repeat sequences
```

```
%allows the user the choice of signal length
repeat_length = 'y';
while repeat_length == 'y'

  length=input('Enter the length of the signal which can only be an even integer between 6 and
49998 inclusive, and not divisible by 4.\n');
  while (length<6) | (length>49998) | ((mod(length,2))~=0) | ((mod(length,4))==0)
    if (length<6) | (length>49998)
      disp ('This is out of the available range.')
    elseif mod(length,2)~=0
      disp ('This is not an even number.')
    else
      disp ('This is divisible by 4.')
    end
    length=input('Please enter an even integer between 6 and 49998 inclusive.\n');
  end

%allows the user the choice of the number of signal levels
  repeat_level = 'y';
  while repeat_level == 'y'

    level = menu ('Choose one of the following:','binary signal','ternary signal');
    half_length = length/2;

%generates binary signals
    if level == 1

%checks if pseudo-random signals are available
      if (MLB_exist(half_length)==1) | (QRB_exist(half_length)==1) |
(TPB_exist(half_length)==1) | (HAB_exist(half_length)==1)
        count = 0;

%prints on the screen the classes of pseudo-random signals available
        disp ('Pseudo-random binary inverse-repeat signal(s) of this length can be based on the
following classes:')

        if MLB_exist(half_length) == 1
          disp ('Maximum length binary signal')
          count = count+1;
          MLB_present = 1;
        else
          MLB_present = 0;
        end
        if QRB_exist(half_length) == 1
          disp ('Quadratic residue binary signal')
          count = count +1;
          QRB_present = 1;
        else
          QRB_present = 0;
        end
        if TPB_exist(half_length) == 1
          disp ('Twin Prime signal')
          count = count+1;
```

```
    TPB_present = 1;
  else
    TPB_present = 0;
  end
  if HAB_exist(half_length) == 1
    disp ('Hall signal')
    count = count+1;
    HAB_present = 1;
  else
    HAB_present = 0;
  end
  fprintf ('\n')

%gives user the choice to continue or abort
  continue = 'a';
  while (continue~='y') & (continue~='n')
    continue = input('Do you want to generate the signal? y/n [y]:','s');
    if isempty(continue)
      continue = 'y';
    end
  end
  if continue == 'y'
    if count>1

%allows user the choice of signal class
      repeat_class = 'y';
        while repeat_class == 'y'
          disp('Choose a signal class by entering the associated character.')
          if MLB_present == 1
            disp ('Maximum length binary signal : m')
          end
          if QRB_present == 1
            disp ('Quadratic residue binary signal : b')
          end
          if TPB_present == 1
            disp ('Twin Prime signal : t')
          end
          if HAB_present == 1
            disp ('Hall signal : h')
          end
          wrong = 1;
          while wrong == 1

            class = input('','s');
            fprintf ('\n')

%generates the pseudo-random signal class chosen
%generates maximum length binary inverse-repeat signal
              if class == 'm'
                if MLB_present == 1
                  MLB_series = MLB(length/2);
                  inverse = inverse_repeat(MLB_series);
                  signal_display (inverse)
```

```
%calculates upper bound in the crosscorrelation without unsystematic errors due to third-order
%nonlinearities
                    R1 = calculate_R1(MLB_series);
                    fprintf('Upper bound in the crosscorrelation without unsystematic errors due
to third-order nonlinearities, R1 is %d.\n\n', R1)

%allows the user to calculate third-order terms
                    answer = 'a';
                    while (answer~='y') & (answer~='n')
                      answer=input('Do you wish to calculate third-order terms? y/n [y]:','s');
                      if isempty(answer);
                        answer = 'y';
                      end
                    end

                    if answer == 'y'
                      fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to
quit.\n', length-1);
                      disp('The integers must be increasing in value and entered in the format
[integer1 integer2].')
                      delay1 = 1;
                      delay2 = 2;

                      while (delay1~=-1) | (delay2~=-1)
                        delay = input('Delays = ');
                        delay1 = delay(1);
                        delay2 = delay(2);
                        while ((delay1<1) | (delay1>length-1) | (delay2<1) | (delay2>length-1) |
(delay1>delay2)) & ((delay1~=-1) | (delay2~=-1))
                          fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to
quit.\n', length-1);
                          disp('The integers must be increasing in value and entered in the
format [integer1 integer2].')
                          delay = input('Delays = ');
                          delay1 = delay(1);
                          delay2 = delay(2);
                        end
                        if (delay1~=-1) & (delay2~=-1)
                          shift = third_shift(MLB_series,delay1,delay2);
                          fprintf('The resulting shift is %d.\n\n', shift)
                        end
                      end
                    end

                    wrong = 0;
                  else
                    wrong = 1;
                  end

%generates quadratic residue binary inverse-repeat signal
                elseif class == 'b'
                  if QRB_present == 1
                    QRB_series = QRB(half_length);
```

```
                inverse = inverse_repeat(QRB_series);
                signal_display (inverse)
                wrong = 0;
              else
                wrong = 1;
              end

%generates Twin Prime inverse-repeat signal
            elseif class == 't'
              if TPB_present == 1
                TPB_series = TPB(half_length);
                inverse = inverse_repeat(TPB_series);
                signal_display (inverse)
                wrong = 0;
              else
                wrong = 1;
              end

%generates Hall inverse-repeat signal
            elseif class == 'h'
              if HAB_present == 1
                HAB_series = HAB(half_length);
                inverse = inverse_repeat(HAB_series);
                signal_display (inverse)
                wrong = 0;
              else
                wrong = 1;
              end
            else
              wrong = 1;
            end

            if wrong == 1
              disp ('Please only enter an available choice.')
            end
          end

          repeat_class = 'a';
          while (repeat_class~='y') & (repeat_class~='n')
            repeat_class = input('Do you want to generate a pseudo-random inverse-repeat
signal based on a different class? y/n [y]:','s');
            if isempty(repeat_class)
              repeat_class = 'y';
            end
          end
        end
      else

%generates maximum length binary inverse-repeat signal
        if MLB_present == 1
          MLB_series = MLB(length/2);
          inverse = inverse_repeat(MLB_series);
          signal_display (inverse)
```

```
%calculates upper bound in the crosscorrelation without unsystematic errors due to third-order
%nonlinearities
            R1 = calculate_R1(MLB_series);
            fprintf('Upper bound in the crosscorrelation without unsystematic errors due to
third-order nonlinearities, R1 is %d.\n\n', R1)
%allows the user to calculate third-order terms
            answer = 'a';
            while (answer~='y') & (answer~='n')
              answer=input('Do you wish to calculate third-order terms? y/n [y]:','s');
              if isempty(answer);
                answer = 'y';
              end
            end

            if answer == 'y'
              fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to quit.\n',
length-1);
              disp('The integers must be increasing in value and entered in the format
[integer1 integer2].')
              delay1 = 1;
              delay2 = 2;
              while (delay1~=-1) | (delay2~=-1)
                delay = input('Delays = ');
                delay1 = delay(1);
                delay2 = delay(2);
                while ((delay1<1) | (delay1>length-1) | (delay2<1) | (delay2>length-1) |
(delay1>delay2)) & ((delay1~=-1) | (delay2~=-1))
                  fprintf('Please enter two integers between 1 and %d or enter [-1 -1] to
quit.\n', length-1);
                  disp('The integers must be increasing in value and entered in the format
[integer1 integer2].')
                  delay = input('Delays = ');
                  delay1 = delay(1);
                  delay2 = delay(2);
                end
                if (delay1~=-1) & (delay2~=-1)
                  shift = third_shift(MLB_series,delay1,delay2);
                  fprintf('The resulting shift is %d.\n\n', shift)
                end
              end
            end

%generates quadratic residue binary inverse-repeat signal
          elseif QRB_present == 1
            QRB_series = QRB(half_length);
            inverse = inverse_repeat(QRB_series);
            signal_display (inverse)

%generates Twin Prime inverse-repeat signal
          elseif TPB_present == 1
            TPB_series = TPB(half_length);
            inverse = inverse_repeat(TPB_series);
            signal_display (inverse)
```

```
%generates Hall inverse-repeat signal
            elseif HAB_present == 1
              HAB_series = HAB(half_length);
              inverse = inverse_repeat(HAB_series);
              signal_display (inverse)
            end
          end
        end

%no pseudo-random binary signals available
      else
        disp ('No pseudo-random binary inverse-repeat signal is available for this length.')
        fprintf ('\n')
      end
    end

%generates ternary signals
    if level == 2
      if (QRT_exist(half_length)==1)
        disp ('Pseudo-random ternary inverse-repeat signal(s) of this length can be based on the
following classes:')
        disp ('Quadratic residue ternary signal')
        fprintf ('\n')

%gives user the choice to continue or abort
        continue = 'a';
        while (continue~='y') & (continue~='n')
          continue = input('Do you want to generate the signal? y/n [y]:','s');
          if isempty(continue)
            continue = 'y';
          end
        end

%generates quadratic residue ternary inverse-repeat signal
        if continue == 'y';
          QRT_series = QRT(half_length);
          inverse = inverse_repeat(QRT_series);
          signal_display (inverse)
        end

%no pseudo-random ternary signals available
      else
        disp ('No pseudo-random ternary inverse-repeat signal is available for this length.')
        fprintf ('\n')
      end
    end

    repeat_level = 'a';
    while (repeat_level~='y') & (repeat_level~='n')
      repeat_level = input('Do you want to generate a signal with a different number of levels?
y/n [y]:','s');
      if isempty(repeat_level)
        repeat_level = 'y';
```

```
      end
    end
  end
  repeat_length = 'a';
  while (repeat_length~='y') & (repeat_length~='n')
    repeat_length = input('Do you want to generate a signal with a different length? y/n [y]:','s');
    if isempty(repeat_length)
      repeat_length = 'y';
    end
  end
end


function QRB_series = QRB(length)

%Calculates the quadratic residue binary series given the signal length.
%
%   QRB_series = QRB(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   QRB_series = Amplitude vector of the generated quadratic residue binary series
%
%   Algorithm:
%     The element r of the signal, a(r) is 1 if r is a square, mod(length).
%     a(r) is -1 otherwise.
%     a(length) is 1.
%
%   Example:
%     QRB_series = QRB(31);

QRB_series = -ones(length,1);
QRB_series(mod([1:(length-1)/2].^2,length))=1;
QRB_series(length)=1;


function exist = QRB_exist(length)

%Checks if the quadratic residue binary sequence exists for a particular length.
%
%   QRB_exist(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   exist = 1 means that the quadratic residue binary sequence exists.
%         = 0 means that the quadratic residue binary sequence does not exist.
%
%   Example:
%     exist = QRB_exist(46);
```

```
if (prime(length)==1) & (mod(length+1,4)==0)
  exist=1;
else
  exist=0;
end
```

function QRT_series = ***QRT***(length)

```
%Calculates the quadratic residue ternary series given the signal length.
%
%   QRT_series = QRT(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   QRT_series = Amplitude vector of the generated quadratic residue ternary series
%
%   Algorithm:
%     The element r of the signal, a(r) is 1 if r is a square, mod(length).
%     a(r) is -1 otherwise. a(length) is 0.
%
%   Example:
%     QRT_series = QRT(31);

QRT_series = QRB (length);QRT_series (length) = 0;
```

function exist = ***QRT_exist***(length)

```
%Checks if the quadratic residue ternary sequence exists for a particular length.
%
%   QRT_exist(length)
%
%   Input argument:
%   length = Length of the signal
%   Output argument:
%   exist = 1 means that the quadratic residue ternary sequence exists.
%         = 0 means that the quadratic residue ternary sequence does not exist.
%
%   Example:
%     exist = QRT_exist(46);

if (QRB_exist(length)==1) | ((mod(length-1,4)==0) & (prime(length)==1))
  exist=1;
else
  exist=0;
end
```

```
function shift = second_shift(bitseries,delay)

%Calculates the delay when a binary sequence is multiplied by the same sequence shifted by a
%certain delay.
%
%   shift = second_shift(bitseries,delay)
%
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   delay = Delay in the second sequence with respect to the first
%
%   Example:
%     x=mlbs(7);
%     second_shift(x,1)

series1 = [bitseries' bitseries'];
%calculates the sequence generated
for n = 1:length(bitseries)
  series2(n) = series1(n+delay); series3(n) = series1(n) * series2(n);
end

found = 0;
%compares the generated sequence with the original sequence
for t = 0:length(bitseries)-1
  for m = 1:length(series3)
    if series3(m) == -series1(m+t)
      if m == length(series3)
        found = 1;
        break
      end
    else
      break
    end
  end
  if found == 1
    shift = length(series3)-t+delay;
    if shift >= length(bitseries)
      shift = shift-length(bitseries);
    end
    break
  end
end


function sequence_plot(bitseries,full_period)

%Plots the signal levels against the clock pulses.
%This function assumes that signals with even signal length have zero mean value.
%The mean is printed only if the signal length is odd.
%   sequence_plot(bitseries,full_period)
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   full_period = Vector of one period of clock pulses plus one extra clock pulse
```

```
%   Example:
%      bitseries = [1; 1; -1; 1; -1; 1];
%      full_period = [1:7];
%      sequence_plot(bitseries,full_period)

full_length = [bitseries; bitseries(length(bitseries))];

%plots the signal levels against the clock pulses
figure(1)
stairs(full_period-1, full_length)
axis([0  length(bitseries)  -1.2  1.2])
title ('Plot of Signal Levels')
xlabel ('Number of Clock Pulses')
ylabel ('Signal Level')
zoom on

%prints the signal mean
if mod(length(bitseries),2)==1
  fprintf('Mean value of the signal is %f.\n\n', mean(bitseries))
end


function signal_display(bitseries)

%Display a given signal on the screen as a table.
%Prints the PIPS, PIPSE and the EMINE of a signal onto the screen.
%Allows the user to save the above data into an associated file.
%Plots the signal, its discrete Fourier transform magnitude, power spectrum and
%autocorrelation.
%
%   signal_display(bitseries)
%
%   Input argument:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   Example:
%   bitseries = [1; 1; -1; -1; 1];
%   signal_display(bitseries)

%the key determines the style of display for the discrete Fourier transform and the
%autocorrelation sections
key = 1;

clock_pulse = 1:length(bitseries);
harmonic = 1:3*length(bitseries);
full_period = 1:length(clock_pulse)+1;
signal = [clock_pulse' bitseries];

%prints the signal onto the screen as a table
disp ('The sequence is')
disp ('  number level')
disp (signal)
```

```
%calculates PIPS
PIPS = calculate_PIPS(bitseries);
fprintf ('PIPS = %f%%\n\n', PIPS)

%calculates PIPSE
if mod(length(bitseries),2)==1
  freqv = 1:(length(bitseries)-1)/2;
else
  freqv = 1:2:length(bitseries)/2-2;
end
PIPSE = calculate_PIPSE(bitseries,freqv);
fprintf ('PIPSE = %f%%\n\n', PIPSE)

%calculates EMINE
if mod(length(bitseries),2)==1
  least_power = (length(bitseries)-1)/2;
  angle = pi*least_power/length(bitseries);
  for n = 1:least_power
    other_angles(n) = pi*n/length(bitseries);
    power(n) = (sin(other_angles(n))/other_angles(n))^2;
  end
  EMINE = 100*sin(angle)/angle/(sqrt(sum(power)/least_power));
else
  least_power = (length(bitseries)/2)-2;
  angle = pi*least_power/length(bitseries);
  for n = 1:2:least_power
    other_angles(n) = pi*n/length(bitseries);
    power(n) = (sin(other_angles(n))/other_angles(n))^2;
  end
  EMINE = 100*sin(angle)/angle/(sqrt(sum(power)/((least_power+1)/2)));
end
fprintf('EMINE = %f%%\n\n', EMINE)

%allows the user to save the data into a separate file called 'perturbation_output'
record  = 'a';
while (record~='y') & (record~='n')
  record=input('Do you wish to save this data? y/n [y]:','s');
  if isempty(record);
    record = 'y';
  end
end

%saves the data into 'perturbation_output'
if record == 'y'
  fid=fopen('perturbation_output','at');
  fprintf (fid,'Pseudo-random signal\n');
  fprintf (fid,'Length = %d\n', length(bitseries));
  fprintf (fid,'\n');
  for row = 1:length(bitseries)
    fprintf (fid,'%d   %d\n',clock_pulse(row),bitseries(row));
  end
  fprintf (fid,'\n');
  fprintf (fid,'PIPS = %f%%\n\n', PIPS);
```

```
  fprintf (fid,'PIPSE = %f%%\n\n', PIPSE);
  fprintf (fid,'EMINE = %f%%\n\n\n', EMINE);
fclose(fid);
end

%allows the user to choose whether to see any plots
answer = 'a';
while (answer~='y') & (answer~='n')
  answer=input('Do you wish to see any plots? y/n [y]:','s');
  if isempty(answer);
    answer = 'y';
  end
end

if answer == 'y'
%plots the signal levels
  sequence_plot(bitseries,full_period)
  fprintf('Press any key to continue.\n\n')
  pause
%plots the discrete Fourier transform magnitude
  DFT_plot(bitseries,clock_pulse,key)
  fprintf('Press any key to continue.\n\n')
  pause
%plots the power spectrum
  power_plot(bitseries,harmonic)
  fprintf('Press any key to continue.\n\n')
  pause
%plots the autocorrelation magnitude
  autocorrelation_plot(bitseries,full_period,key)
end


function shift = third_shift(bitseries,delay1,delay2)

%Calculates the delay when a binary sequence is multiplied twice by the same sequence shifted
%by a two different delays.
%
%   shift = third_shift(bitseries,delay1,delay2)
%
%   Input arguments:
%   bitseries = Vector containing the amplitudes of the signal sequence
%   delay1 = Delay in the second sequence with respect to the first
%   delay2 = Delay in the third sequence with respect to the first
%
%   Output argument:
%   shift = Delay in the output sequence
%
%   Example:
%     x=mlbs(7);
%     shift = third_shift(x,1,2)

series1 = [bitseries' bitseries'];
```

```
%calculates the sequence generated
for n = 1:length(bitseries)
  series2(n) = series1(n+delay1); series3(n) = series1(n) * series2(n);
  series4(n) = series1(n+delay2); series5(n) = series3(n) * series4(n);
end

found = 0;
%compares the generated sequence with the original sequence
for t = 0:length(bitseries)-1
  for m = 1:length(series5)
    if series5(m) == series1(m+t)
      if m == length(series5)
        found = 1;
        break
      end
    else
      break
    end
  end
  if found == 1
    shift = length(series5)-t+delay2;
    break
  end
end


function TPB_series = TPB(length)

%Calculates the Twin Prime sequence given the signal length.
%
%   TPB_series = TPB(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   TPB_series = Amplitude vector of the generated Twin Prime sequence
%
%   Algorithm:
%     Twin Prime sequence exists for length, N = p*q
%     where q = p+2.
%     A quadratic residue binary sequence, {a(r)} is formed for length p.
%     A quadratic residue binary sequence, {b(r)} is formed for length q.
%
%     The Twin Prime sequence, {c(r)} is defined as follows:
%       c(r) = 1 for r=0 (mod(q))
%       c(r) = -1 for r=0 (mod(p)) but r~=0 (mod(q))
%       c(r) = a(r)*b(r) otherwise
%
%   Example:
%     TPB_series = TPB(31);

%calculates the smaller root
```

```
root = -1+sqrt(1+length);
%generates the quadratic residue sequence of length equal to the smaller root
series = QRB(root);
for k = 0:root:length-root
  for n = 1:root
    QRB_root(n+k) = series(n,:);
  end
end
%calculates the larger root
root2 = root+2;
%generates the quadratic residue sequence of length equal to the larger root
series2 = QRB(root2);
for k2 = 0:root2:length-root2
  for n2 = 1:root2
    QRB_root2(n2+k2) = series2(n2,:);
  end
end
%generates the Twin Prime sequence
TPB_series=ones(length,1);
for number = 1:length
  if mod(number,root2)==0
    TPB_series(number) = 1;
  elseif (mod(number,root)==0) & (mod(number,root2)~=0)
    TPB_series(number) = -1;
  else
    TPB_series(number) = QRB_root(number) * QRB_root2(number);
  end
end


function exist = TPB_exist(length)

%Checks if the Twin Prime sequence exists for a particular length.
%
%   TPB_exist(length)
%
%   Input argument:
%   length = Length of the signal
%
%   Output argument:
%   exist = 1 means that the Twin Prime sequence exists.
%         = 0 means that the Twin Prime sequence does not exist.
%
%   Example:
%     exist = TPB_exist(46);

root = -1+sqrt(1+length);
if (prime(root)==1) & (prime(root+2)==1) & (mod(root,1)==0)
  exist=1;
else
  exist=0;
end
```